The Level Set Method was introduced in the seminal paper of Osher and Sethian [44] in 1988; however some of the ideas involved already existed in the literature; see e.g. [23]. It is now acknowledged as one method of choice when dealing with arbitrarily large motions of domains or surfaces (even ones involving topological changes), from both theoretical and numerical perspectives.

This chapter is a modest and inevitably biased introduction to the basic features of this method, without any ambition of exhaustivity. It is mainly focused on its practical aspects: in order to keep the exposition elementary, on several occasions, we chose not to dig into the (very deep) underlying mathematical theory.
This chapter is organized as follows. Section 1 sets the scope of the study by presenting the types of evolving domains or surfaces we shall get interested in. In Section 2, the level set framework is introduced, namely the key idea of describing domains in an implicit way, via an auxiliary ‘level set’ function. Subsections 2.2 and 2.4 deal in a formal way with the connection between the ‘intuitive’ notion of an evolving domain, and the corresponding evolution equations for an associated level set function; Subsection 2.3 attempts to provide a glimpse of the underlying mathematical theory; it may be omitted at first reading, or by the uninterested reader.

In Section 3, we describe a typical use of the Level Set Method in the context of the resolution of a concrete physical problem, without yielding to technicality. It is emphasized that, in numerical practice, the Level Set Method relies on two main operations. Section 4 discusses numerical algorithms devoted to the first of these, namely the resolution of the level set evolution equation. Then, Section 5 presents numerical methods for the second operation of interest, - that of creating one level set function associated to a given domain or surface ∂Ω, among which the celebrated Fast Marching Method.

Several operations which are easy to perform when a conventional representation of domains is used may result slightly more difficult in the level set framework; in Section 6, we discuss numerical methods for the main such operations. Section 7 is then a pot-pourri of additional practical issues: the need for redistancing is discussed, as well as the Narrow Band improvement, etc.

Numerical methods for tracking the evolution of domains are manifold. Section 8 briefly presents other popular methods, together with their main differences with the Level Set Method.

We conclude this overview of the Level Set method with Section 9, in which we reference several applications where it has been successfully used, e.g. in numerical simulation, image processing, etc.

Before entering the core of the matter, let us mention that very good in-depth monographs exist on the topic: the obvious and perhaps most complete reference is the book of Sethian [54]; see also that of Osher and Fedkiw [43]. These may be fruitfully complemented with the more theoretic monograph [28].

1. Generalities about domain, or surface evolution problems

The Level Set Method is a general framework for the theoretical and numerical study of an evolving domain Ω(t) ⊂ Rd (or equivalently its boundary Γ(t) := ∂Ω(t)) according to a velocity field (t,x) ↦ V(t,x) ∈ Rd. The velocity field V may account for motions of different natures, which can roughly be classified into three categories:

1. V may be externally prescribed, that is, independently from any feature of the domain Ω(t). For instance, one may imagine that Ω(t) is a small object, passively transported in a fluid with velocity V. Of course, this coarse model overlooks any physical interaction between the object and the fluid.

2. V may depend on local features of Ω(t) or Γ(t), such as the normal vector n(t)(x) (pointing outward Ω(t)), or the mean curvature κ(t)(x) of Γ(t). Let us present two classical examples of such motions, which will be frequently used throughout this chapter:
   • In the model of propagation of a flame front, studied in [9, 52], Ω(t) stands for a burnt region, whose frontier Γ(t) expands with constant, normal velocity c > 0. In other words, V reads:
   \[
   V(t,x) = c \cdot n(t)(x).
   \]
   • The mean curvature flow features the velocity field
   \[
   V(t,x) = -κ(t)n(t)(x).
   \]
   Intuitively, Γ(t) evolves by resorption of its ‘bumpy’ regions (i.e. the regions where its mean curvature is positive), and ‘corking’ of its ‘creases’ (the regions where its mean curvature is negative). More rigorously, we have seen in Chapter ?? that the mean curvature flow arises as the gradient flow associated to the minimization of the perimeter functional.

3. V may depend on global features of Ω(t) or Γ(t). For instance, when Γ(t) describes the interface between two fluids with different physical properties, V arises as V(t, ·) = u(Ω(t)), the solution to the bифluid Navier-Stokes equations (see Section 3, or Chapter ??).

\[^1\]The second fundamental form Πx (resp. the mean curvature κ(x)) of the boundary Γ of Ω is oriented in the sense that it is positive definite (resp. positive) when Ω is locally convex around x.
This last class of motions is undoubtedly the most crucial one from a physical viewpoint; unfortunately, they generally prove far too complicated to lend themselves to a rigorous analysis. Therefore, most of this chapter deals with motions of the first two kinds. As we shall see in Section 3, this is not a great loss of generality, since most of the methods for the study of this third kind of motions are somehow reduced to one (or several ones) of the first two kinds.

2. The Level Set Method for describing domain or interface evolution

The Level Set Method relies on the representation of a domain $\Omega \subset \mathbb{R}^d$ in implicit form, that is, via a scalar ‘level set’ function $\phi : \mathbb{R}^d \rightarrow \mathbb{R}$ satisfying the properties (see Figure 1 for an illustration):

$$\begin{cases} 
\phi(x) < 0 & \text{if } x \in \Omega, \\
\phi(x) = 0 & \text{if } x \in \Gamma := \partial \Omega, \quad x \in \mathbb{R}^d, \\
\phi(x) > 0 & \text{if } x \in \mathbb{R}^d. 
\end{cases}$$

(2.1)

In other words, $\Omega$ is the negative subdomain of $\phi$, and $\Gamma$ matches with its 0 isovalue. Note that there exist ‘many’ (actually, an infinity) such level set functions associated to the same domain $\Omega$. As we shall see soon, this change in perspectives allows for a very convenient formulation of the evolution of a domain. Before getting into details, let us first discuss about how geometric quantities attached to $\Omega$ may be directly calculated from this function $\phi$.

2.1. Level Set framework and geometry.

Let $\Omega \subset \mathbb{R}^d$ be a bounded domain of class $C^1$, $\phi : \mathbb{R}^d \rightarrow \mathbb{R}$ be an associated level set function (i.e. (2.1) holds), such that $\nabla \phi \neq 0$ on a neighborhood of $\Gamma$. Note that such a function always exists, as a consequence of Definition ?? of a domain of class $C^1$, and of a classical argument of partitions of unity.

The unit normal vector $n(x)$ to $\Gamma$, pointing outward $\Omega$ can be expressed in terms of $\phi$ as:

$$\forall x \in \Gamma, \quad n(x) = \frac{\nabla \phi(x)}{|\nabla \phi(x)|}.$$

(2.2)

Assuming $\Omega$ and $\phi$ to be addtionally of class $C^2$, the second fundamental form $\Pi$ associated to $\Gamma$ has the expression:

$$\forall x \in \Gamma, \quad \Pi_x = \nabla \left( \frac{\nabla \phi(x)}{|\nabla \phi(x)|} \right).$$

(2.3)
In particular, the mean curvature $\kappa$ of $\Gamma$, that is, the trace of $\Pi$ reads:

$$
\forall x \in \Gamma, \quad \kappa(x) = \text{div} \left( \frac{\nabla \phi(x)}{|\nabla \phi(x)|} \right).
$$

2.2. Surface evolution in the level set framework.

We have hitherto been purposely evasive about what we precisely mean by ‘domain evolution’. In order to appraise the difficulties inherent to this notion, let us start by a formal discussion around a model situation: $\Omega(t) \subset \mathbb{R}^d$ is a smooth bounded domain, evolving over a time period $(0, T)$, according to a smooth velocity field $V(t, x)$, defined for $t \in (0, T)$ and $x \in \mathbb{R}^d$.

By this, it is natural to understand that, for any two times $t_0, t \in (0, T)$, $\Omega(t)$ may be obtained from $\Omega(t_0)$ by transporting its points along $V$ (see Figure 2):

$$
\Omega(t) = \{ \chi(x_0, t, t_0), \; x_0 \in \Omega(t_0) \},
$$

where $t \mapsto \chi(x_0, t, t_0)$ is the integral curve of $V$ emerging from $x_0$ at time $t_0$, the unique solution of the ordinary differential equation:

$$
\left\{ \begin{array}{l}
\frac{d}{dt}(\chi(x_0, t, t_0)) = V(t, \chi(x_0, t, t_0)) \text{ for } t > 0, \\
\chi(x_0, t_0, t_0) = x_0.
\end{array} \right.
$$

Recall that we have already encountered this notion in the context of fluid mechanics (Definition ??): $t \mapsto \chi(x_0, t, t_0)$ is the characteristic curve emerging from $x_0$ at time $t_0$; it describes the position of a particle flowing along the velocity field $V(t, x)$, which lies at $x_0$ at time $t_0$.

![Figure 2. One domain $\Omega(t)$ evolving according to the velocity field $V(t, x)$.](image)

Now, for each time $t$, let $\phi(t, \cdot)$ be a smooth level set function for $\Omega(t)$, and let us see how the motion of $\Omega(t)$ translates in terms of $\phi$. Let $t_0 \in (0, T)$ and $x_0 \in \Gamma(t_0)$ be given, and let $t \mapsto \chi(x_0, t, t_0)$ be the trajectory defined by (2.5). It stems from the definitions that:

$$
\forall t \in (0, T), \quad \phi(t, \chi(x_0, t, t_0)) = 0;
$$
a direct application of the chain-rule then yields:

$$
\frac{d}{dt}(\phi(t, \chi(x_0, t, t_0))) = \frac{\partial \phi}{\partial t}(t, \chi(x_0, t, t_0)) + \chi'(x_0, t, t_0) \cdot \nabla \phi(t, \chi(x_0, t, t_0)) = 0.
$$

Evaluating this identity at $t = t_0$, and since the argument holds for any $t_0 \in (0, T)$, $x_0 \in \Gamma(t_0)$, we finally obtain the so-called level set advection equation:

$$
\forall t \in (0, T), \; x \in \mathbb{R}^d, \quad \frac{\partial \phi}{\partial t}(t, x) + V(t, x) \cdot \nabla \phi(t, x) = 0.
$$
Note that (2.6) is not a usual advection equation, since \( V \) generally depends on \( \Omega \) - thus on \( \phi \), except in very specific cases (see the discussion in Section 1).

At this point, several comments are in order:

1. Strictly speaking, we have only proved (2.6) for pairs \((t, x) \in (0, T) \times \mathbb{R}^d\) such that \( x \in \Gamma(t) \). However, imposing that every level set of \( \phi(t, \cdot) \) (and not only that associated to the value 0) evolves according to \( V(t, \cdot) \), the previous argument leads precisely to (2.6).

2. The normal vector \( n_t(x) \) to \( \Gamma(t) \) at \( x \), pointing outward \( \Omega(t) \), reads \( n_t(x) = \frac{\nabla \phi(t, x)}{|\nabla \phi(t, x)|} \). Let \( v = V \cdot \frac{\nabla \phi}{|\nabla \phi|} \) be the normal component of \( V \); (2.6) then rewrites as the so-called level set Hamilton-Jacobi equation:

\[
\forall t \in (0, T), \ x \in \mathbb{R}^d, \ \frac{\partial \phi}{\partial t}(t, x) + v(t, x)|\nabla \phi(t, x)| = 0. \tag{2.7}
\]

Again, this terminology is a bit misleading since, in general, (2.7) is not a Hamilton-Jacobi equation.

3. Equation (2.7) expresses the intuitive fact that only the normal component of \( V \) plays a role in the evolution of \( \Omega(t) \): its tangential part only accounts for a ‘convection’, or reparametrization. Note that the same phenomenon is at play when it comes to the structure of shape derivatives (see Chapter ??, Th. ??).

4. The Level Set Method requires the velocity field \( V(t, \cdot) \) to be defined on the whole ambient space \( \mathbb{R}^d \), whereas, in many applications, it only makes sense on \( \Gamma(t) \) (see the examples given in the second and third items in Section 1). When such is the case, \( V(t, \cdot) \) must be extended to \( \mathbb{R}^d \) (or at least a neighborhood of \( \Gamma(t) \)). It is however not difficult to show (at least formally) that the evolution of the 0 level set of \( \phi(t, \cdot) \) (and only the 0 level set) does not depend on how this velocity is extended. We will see in Section 7.2 that the same necessity also arises in the numerical context.

The Level Set method is thus a convenient rephrasing of the motion of \( \Omega(t) \): this difficult geometric evolution problem translates into the PDE (2.6-2.7), posed in terms of the auxiliary function \( \phi(t, \cdot) \), which is often simpler to handle - especially from the numerical point of view. This reparametrization allows to account for rather arbitrary deformations of \( \Omega(t) \), including changes in its topology (for instance, merging of some holes); these are indeed naturally addressed in this framework, and would be very difficult to describe with a conventional representation of domains. Consider the example depicted in Figure 3: a domain \( \Omega(t) \) initially composed of two disks evolves by ‘inflating’ until its two components merge. The instant where \( \Omega(t) \) changes topology is very awkward to describe with a conventional description of domains, and fairly natural with a level set description. We shall say more about this point in the next section.

2.3. A glimpse at the mathematical framework.

However appealing, the considerations of the previous section are formal. To be more precise, they are legitimate as long as the evolving domain \( \Omega(t) \) and the velocity field \( V(t, x) \) stay smooth. Unfortunately, even in the context of a simple motion, intitiated with an arbitrarily smooth domain \( \Omega(0) \), \( \Omega(t) \) turns out to develop singularities. Let us illustrate this important feature with two examples.

1. This first example deals with the flame propagation model, presented in Section 1, and is excerpted from [54] \S2.3. The situation is that of Figure 4: a smooth two-dimensional domain \( \Omega(0) \), whose boundary \( \Gamma(0) \) is locally described by the curve

\[
[0, 1] \ni s \rightarrow \gamma(s) = \left( 1 - s, \ \frac{1 + \cos(2\pi s)}{2} \right) \in \mathbb{R}^2,
\]

accounts for the initially burnt region. The flame front \( \Gamma(t) \) expands with unit normal velocity \((v(t, x) = n_t(x) \) in the notation of the previous section\), and several positions of the front are depicted. At some time \( t = t_c \), \( \Gamma(t) \) develops a singularity: it is no longer smooth at the blue dot on Figure 4.

2. Let us consider the mean curvature flow (1.2), with the ‘dumbbell’ of Figure 5 (left) as initial domain. It is observed in [17] that \( \Omega(t) \) evolves by shrinking, until two ends of its boundary \( \Gamma(t) \) join, as in Figure 5 (right).
Figure 3. Inflation of a domain $\Omega(t)$ initially composed of two disconnected disks; the upper row shows the motion of the boundary $\Gamma(t)$, and the lower one, the corresponding evolution of the graph of a corresponding level set function $\phi(t,\cdot)$.

Figure 4. Representation of $\Omega(t)$ for $t = 0, 0.02, 0.04, \text{ and } t = 0.055$ (from bottom to top), in the flame propagation model. The initially burnt region $\Omega(0)$ (in grey) is of class $C^\infty$; however, a singularity (blue dot) develops at approximately $t_c = 0.055$.

After $\Omega(t)$ has developed a singularity, it is unclear how its motion should go on; in both examples discussed above, the normal vector field $n_t$ and mean curvature $\kappa_t$ of $\Gamma(t)$, are no longer everywhere defined.
Figure 5. Evolution of a three-dimensional dumbbell under the mean curvature flow. The central part of the bar ends up pinching.

What is the ‘correct’ evolution of Ω(t) for t > t_c is actually a matter of definition, and is specific to every particular situation. Let us elaborate on the first example discussed above: the perhaps most ‘natural’ way to characterize the motion of Ω(t) for t > t_c is to demand that all the points x in Γ(t) at which the normal vector n_t(x) is well-defined go on moving independently with a unit normal velocity. The subsequent evolution of Γ(t) is drawn on Figure 6 (left): Γ(t) is now a self-intersecting curve. Another way to define this evolution, which relies on the physical meaning of the problem, is to impose an ‘entropy criterion’, according to which a point burnt at some time t_0 stays burnt for t > t_0. In more mathematical terms, for any two times t_1 ≤ t_2, one has Ω(t_1) ⊂ Ω(t_2). The resulting evolution of Γ(t) is represented on Figure 6 (right).

Figure 6. (Left) The ‘swallowtail’ pattern and (right) the ‘cusp’ pattern described by Ω(t), for t > t_c in the flame propagation model.

Imposing a ‘natural’ behavior to Ω(t) once singularities have appeared is not simple in the general case. At first, attempts were made to classify all the singularities which may arise in the course of such evolutions, but this task turned out to be out of reach (especially in the three-dimensional context).

An efficient means to achieve this goes the other way around, and starts from the level set equations (2.6-2.7): the idea is to endow them with a generalized notion of ‘weak’ solutions, that somehow ‘encompasses some physics’; if φ(t, x) is such a weak solution, Ω(t) is then defined as the negative subdomain:

\[ \Omega(t) := \{ x \in \mathbb{R}^d, \, \phi(t, x) < 0 \} . \]

An adapted framework to many such problems turns out to be that of viscosity solutions, initially proposed by P.-L. Lions and M. G. Crandall in [19], in a broader context than that of Equations of the form (2.6-2.7); the interested reader may find a comprehensive introduction to this rich theory in [18].

Definition 1. Let U ⊂ \mathbb{R}^d be an open set and H : \mathbb{R}^d × \mathbb{R}^d × \mathbb{R}^d × S(\mathbb{R}^d) be a continuous function (the Hamiltonian). Consider the general second-order Hamilton-Jacobi equation posed on (0, T) × U:

\[
\frac{\partial \phi}{\partial t}(t, x) + H(x, \phi, \nabla \phi, \nabla^2 \phi)(t, x) = 0,
\]

where \nabla^2 \phi stands for the Hessian matrix of the unknown \phi.
• A function \( \phi \) is a viscosity subsolution of (2.8) if it is upper semicontinuous on \( U \), and, for any function \( \varphi \) of class \( C^2 \) on \( U \) such that \( \phi - \varphi \) reaches a local maximum at \( x \),
\[
\frac{\partial \phi}{\partial t}(t,x) + H(x,\phi(x),\nabla \varphi(x),\nabla^2 \varphi(x)) \leq 0.
\]

• A function \( \phi \) is a viscosity supersolution of (2.8) if it is lower semicontinuous on \( U \), and, for any function \( \varphi \) of class \( C^2 \) on \( U \) such that \( \phi - \varphi \) reaches a local minimum at \( x \),
\[
\frac{\partial \phi}{\partial t}(t,x) + H(x,\phi(x),\nabla \varphi(x),\nabla^2 \varphi(x)) \geq 0.
\]

• \( \phi \) is a viscosity solution of (2.8) if it is both a viscosity subsolution and a viscosity supersolution.

**Remark 1.** Equations of the form (2.6-2.7) directly fall into the framework of the above definition when either the velocity \( V \), or the normal velocity \( v \) is independent of \( \phi \). Such is not the case of the mean curvature flow equation (1.2), which demands an adaptation of the notion of viscosity solution [24].

Let us presently list several properties of the viscosity solution \( \phi \) of an equation of the form (2.8), or of the associated negative subdomain \( \Omega(t) = \{ x \in \mathbb{R}^d, \phi(t,x) < 0 \} \). These properties attest to the ‘physical’ meaning of the notion of viscosity solution; note that they are hereafter formulated in a deliberately hazy way, omitting the (very technical) assumptions. Let us solely mention that they hold for fairly general Hamiltonian functions \( H \) (see [28], Chap. 4 for precise statements and proofs).

- **Existence and unicity.** The viscosity solution \( \phi \) to (2.8) exists and is unique.
- **Generalization of the notion of classical solution.** If \( \phi \) is of class \( C^2 \), then it is also a solution of (2.8) in the classical sense.
- **Vanishing viscosity limit of solutions to ‘regular’ equations.** For small \( \varepsilon > 0 \), let \( \phi_\varepsilon(t,x) \) be the smooth solution to the equation:
\[
\begin{cases}
\frac{\partial \phi_\varepsilon}{\partial t}(t,x) - \varepsilon \Delta \phi_\varepsilon(t,x) + H(x,\phi_\varepsilon,\nabla \phi_\varepsilon,\nabla^2 \phi_\varepsilon)(t,x) = 0, \\
\phi_\varepsilon(t = 0, \cdot) = \phi(t = 0, \cdot)
\end{cases}
\]

obtained from (2.8) by adding the regularizing ‘vanishing viscosity’ term \( -\varepsilon \Delta \phi_\varepsilon \). Then, \( \phi_\varepsilon \xrightarrow{\varepsilon \to 0} \phi \), uniformly on every compact subset of \([0,T] \times \mathbb{R}^d \).

- **Independence from the level set function characterizing the initial domain.** Let \( \Omega_0 \) be a domain, and \( \phi_0, \psi_0 \) be two associated level set functions (i.e. (2.1) holds). Let \( \phi(t, \cdot), \psi(t, \cdot) \) be the solutions of (2.8) associated to the respective initial data \( \phi_0, \psi_0 \). Then, the domains described by \( \phi \) and \( \psi \) coincide, i.e.
\[
\{ x \in \mathbb{R}^d, \phi(t,x) < 0 \} = \{ x \in \mathbb{R}^d, \psi(t,x) < 0 \}.
\]

- **Monotonicity.** Let \( \Omega_0 \subset \overline{\Omega}_0 \) be two domains of \( \mathbb{R}^d \), and \( \Omega(t), \tilde{\Omega}(t) \) be the evolving domains resulting from the process described above; i.e. \( \Omega(t) \) (resp. \( \tilde{\Omega}(t) \)) is the negative subdomain of \( \phi(t, \cdot) \) (resp. \( \tilde{\phi}(t, \cdot) \)), the solution to (2.8) with an initial data which is a level set function for \( \Omega(0) \) (resp. \( \tilde{\Omega}(0) \)). In this context, \( \Omega(t) \subset \tilde{\Omega}(t) \).

Before closing this section, let us appraise the concrete meaning of this notion of solutions on the two examples introduced above.

- As was proved in [9], the evolution of \( \Omega(t) \) in the flame propagation model after the first singularity has appeared is that obtained by imposing the so-called ‘entropy criterion’ (see Figure 4, right).
- In the case of the mean curvature flow, the two components joined by the singular point separate, and each part shrinks independently to a point. Other interesting examples of domains evolving via the mean curvature flow can be found in [14, 24] and the references therein.

### 2.4. Domain evolution as a boundary value problem: Eikonal equations.

An interesting particular case of the previous considerations arises when \( \Omega(t) \) expands according to a normal velocity, i.e. \( V(t,x) \) is of the form:
\[
V(t,x) = c(x)n_t(x),
\]

[8]
with \( c(x) > 0 \); this holds in particular in the example of flame propagation discussed above. The problem turns out to be equivalently described by the stationary time function \( T : \mathbb{R}^d \setminus \overline{\Omega(0)} \to \mathbb{R} \), defined as:

\[
T(x) = \inf \{ t \geq 0, \ x \in \Omega(t) \},
\]

that is, for any \( x \in \mathbb{R}^d \setminus \overline{\Omega(0)} \), \( T(x) \) is the first time \( t \) at which \( \Omega(t) \) reaches \( x \).

The derivation of a boundary value problem for \( T \) follows the same trail as that of the level set equations (2.6-2.7): in a first step, it is rigorously established in regions where all the quantities at stake are smooth enough. Then, \( T \) is understood as the solution of this partial differential equation in the whole ambient space, in an adequately generalized sense.

Let \( x_0 \in \mathbb{R}^d \), and assume that the functions \( T, c \) are smooth in a vicinity \( U \) of \( x_0 \). Using again the intuitive notion of an evolving domain of Section 2.2, let \( x(t) := \chi(x_0, t, t_0) \) be the integral curve of \( V \) emerging from \( x_0 \) at \( t = t_0 \), which is defined on a neighborhood of \( t_0 \) by (2.5); recall that it satisfies the properties:

\[
x(t_0) = x_0, \ x(t) \in \Gamma(t) \text{ and } x'(t) = c(x(t))n_t(x(t)).
\]

On the other hand, it follows from the very definition of \( T \) that:

\[
\Omega(t) = \{ x \in \mathbb{R}^d, \ T(x) < t \}, \ \Gamma(t) = \{ x \in \mathbb{R}^d, \ T(x) = t \};
\]

as a consequence, a level set function associated to \( \Omega(t) \) is

\[
\forall (t,x) \in [0,T] \times \mathbb{R}^d, \ \phi(t,x) := T(x) - t,
\]

whence, from formula (2.2), \( n_t(x) = -\frac{\nabla \phi(t,x)}{|\nabla \phi(t,x)|} \). Differentiating with respect to \( t \) in the relation \( T(x(t)) = t \) and incorporating the Dirichlet boundary condition \( T(x) = 0 \), for all \( x \in \Gamma(0) \), we eventually end up with the Eikonal equation:

\[
(2.9) \begin{cases}
c|\nabla T| = 1 & \text{in } \mathbb{R}^d \setminus \overline{\Omega(0)} \\
T = 0 & \text{on } \Gamma(0)
\end{cases}.
\]

Actually, in the sequel, we will also get interested in the very similar case where \( \Omega(t) \) shrinks according to a normal velocity \( c \) (or \( \mathbb{R}^d \setminus \overline{\Omega(t)} \) expands with velocity \( c \)):

\[
V(t,x) = -c(x)n_t(x),
\]

with \( c(x) > 0 \). A similar argument reveals that the associated time function \( T : \Omega(0) \to \mathbb{R} \) is solution to the Eikonal equation:

\[
(2.10) \begin{cases}
c|\nabla T| = 1 & \text{in } \Omega(0) \\
T = 0 & \text{on } \Gamma(0)
\end{cases}.
\]

So as to select unambiguously a ‘physical’ behavior for \( T \), solutions to (2.9) or (2.10) have to be taken into account in a generalized setting, which once again happens to be that of viscosity solutions. Adapting Definition 1 to the present stationary setting (we omit the details, referring again to [18]), the result of interest is now the following (see e.g. [8]):

**Theorem 1.** Assume \( \Omega(0) \subset \mathbb{R}^d \) is a bounded domain, and the normal velocity function \( c \) is positive and uniformly continuous on \( \Omega(0) \). Then, there exists a unique viscosity solution to the Dirichlet problem (2.10).

**Example 1.** Let us briefly look into the interesting particular case of equation (2.10) when a unit normal velocity field, \( c \equiv 1 \) is considered. The unique viscosity solution to (2.10) is the Euclidean distance function \( d(\cdot, \Gamma) \) to \( \Gamma \), defined as:

\[
\forall x \in \Omega, \ d(x, \Gamma) = \inf_{y \in \Gamma} |x - y|.
\]

This fact translates the regular spacing out of the level sets of distance functions, as can be seen on figure 7. We will discuss this very important property once again in Section 5. See also the discussion in [49] Chap. 2, about the meaning of the viscosity solution in this particular case.
As a preliminary to our discussion of the applications of the Level Set Method in scientific computing, we describe a model problem which we expect to tackle in the level set framework. We seize this opportunity to motivate the forthcoming developments of dedicated algorithms - and also to present a challenging real-life application where the Level Set method finds a natural application. In the next sections, we shall often refer to this example to illustrate our purposes.

Let \( D \subset \mathbb{R}^d \) be a computational domain, divided into two phases \( \Omega_0 \) and \( \Omega_1 := D \setminus \Omega_0 \), filled with different fluids with respective (constant) densities and dynamic viscosities \( \rho_0, \rho_1 \) and \( \nu_0, \nu_1 \). The interface between the fluids is then \( \Gamma := \partial \Omega_0 \setminus \partial D \); see Figure 8. The domains \( \Omega_0, \Omega_1 \) are moving in time, and we aim at solving the bifluid Navier-Stokes equations over a time period \((0, T)\) in this context, that is:

\[
\begin{align*}
\rho_i \left( \frac{\partial u_i}{\partial t} + u_i \cdot \nabla u_i \right) - \nu_i \Delta u_i + \nabla p &= f_i \quad \text{for } (t, x) \in (0, T) \times \Omega_i(t), \\
\text{div}(u_i) &= 0 \quad \text{for } (t, x) \in (0, T) \times \Omega_i(t), \\
u_i(t, x) &= 0 \quad \text{for } (t, x) \in (0, T) \times \partial D, \\
\langle \sigma_0 - \sigma_i \rangle \cdot n_t &= \gamma \kappa_t n_t \quad \text{on } \Gamma(t), \\
u_i(t = 0, \cdot) &= u_i(0, \cdot) \quad \text{on } \Omega_i(0)
\end{align*}
\]

where \( n_t(\cdot) \) (resp. \( \kappa_t(\cdot) \)) is the unit normal vector to \( \Gamma(t) \), pointing outward \( \Omega_0(t) \) (resp. its mean curvature), \( f_i \) stands for the applied body force (typically, gravity), and \( \gamma \) is the surface tension coefficient. The interface \( \Gamma \) is consistently moving according to the velocity field \( u(t, \cdot) \) (recall that \( u_0(t, \cdot) \) and \( u_1(t, \cdot) \) coincide on \( \Gamma(t) \)), and we assume that the initial domains and interface \( \Omega_i(0), i = 1, 2, \) and \( \Gamma(0) \) are given.

This example of a moving domain, whose physical and mathematical aspects are broached in Chapter ??, is one among many where the velocity field depends on global features of this domain. Admittedly, this model is already a very complicated one, but the description of this section will not rely on any sort of related theoretical or numerical analysis.

Let us first sketch the considered numerical setting.

**Notations:** The time interval \((0, T)\) is split into subintervals \((t^n, t^{n+1})\) of ‘small’ length \( \Delta t = t^{n+1} - t^n \), where \( n = 0, ..., T/\Delta t \). When \( \phi(t, x) \) is a quantity depending on time and space, \( \phi^n(x) \) stands for the sought numerical approximation of \( \phi(t^n, x) \), for any \( x \in D \).

As for the spatial discretization, the computational domain \( D \) is discretized with a mesh \( \mathcal{T} \), which we do not specify for the time being; for instance, it may be a Cartesian grid, or a simplicial mesh (for the precise meanings of these terms, we refer to Chapter ??). This mesh comes with a set of degrees of freedom, which we assume to be the vertices \( \{ x_i \}_{i \in I} \) of \( \mathcal{T} \) for simplicity.
Figure 8. Model situation for the bifluid problem (3.1).

The two phases $\Omega_0(t), \Omega_1(t)$ (and thus the interface $\Gamma(t)$) are tracked by means of a Level Set function $\phi(t, \cdot)$ for the domain $\Omega_0(t)$ (i.e. (2.1) holds with $\Omega_0(t)$ instead of $\Omega$). This function is, at all times, numerically discretized on the vertices of $\mathcal{T}$. Hence, the quantity of interest when it comes to $\phi$ are the $\phi^n(x_i)$, for $n = 0, \ldots, N$ and $i \in I$.

A tentative outline of an algorithm for solving the considered bifluid flow problem reads as follows:

- **Initialization:** Create a level set function $\phi^0$ (at the nodes of mesh $\mathcal{T}$) associated to the initial domain $\Omega_0(0)$.
- **Main loop:** for $n = 0, \ldots, N - 1$,
  1. **Motion of the domain:** solve the level set advection equation (2.6) (or the level set Hamilton-Jacobi equation (2.7) over the time period $(t^n, t^{n+1})$, with vector velocity $u^n$ (resp. with scalar, normal velocity $u^n \cdot n$), and initial condition $\phi^n$.
     Implicitly, this relies on the assumption that the time step $\Delta t$ is small enough so that the velocity of the fluid $u(t, \cdot)$ may be ‘frozen’ over $(t^n, t^{n+1})$:
     $$\forall t \in (t^n, t^{n+1}), \quad x \in D, \quad u(t, x) \approx u^n(x).$$
  2. **Resolution of the partial differential equation:** Solve the Navier-Stokes Bifluid equation (3.1) over the time period $(t^n, t^{n+1})$ to obtain the new velocity $u^{n+1}$.

This discussion highlights the general needs for numerical methods associated to the Level Set Method:

1. We need an algorithm for generating a level set function $\phi^0$ associated to the initial domain $\Omega_0(0)$. In Section 5, we present efficient numerical schemes to carry out this initialization step.
2. We also need to solve the level set advection equation (2.6), or the Level Set Hamilton-Jacobi equation (2.7) over a given time period, and with given initial value and (scalar or vector) velocity field. This issue is tackled in Section 4.
3. The last main interrogation is about the resolution of the Navier-Stokes bifluid equation (3.1). This step is generally hard to perform, partly because of the complexity of the equation, partly because of the additional difficulty entailed by its coupling with a time-dependent geometry (and thus not specific to the use of the Level Set Method). Techniques for tackling this issue are specific to each equation, and getting into details goes beyond the scope of this section; to get an insight of how it is done in the context of Equation (3.1), we refer to Chapter ??.

Still, whatever these methods, there are operations that we certainly need to perform on a domain $\Omega$ which is only known through an associated level set function $\phi$. Here are some of them:

- Calculate the normal vector $n$ and the curvature $\kappa$ of $\Omega$ (which, in the example above, appear in the transmission conditions in (3.1)),
- Evaluate the integral of a given function over $\Omega$ or $\Gamma$ (in our context, this is needed for assembling the stiffness matrix associated to (3.1)).

These operations will be dealt with in Section 6.
The efficiency, robustness, and relative simplicity in terms of programming effort, of the numerical methods for carrying out the aforementioned operations contribute to some significant degree to the popularity of the Level Set Method. It is a fast and accurate way to account for arbitrarily large deformations of a domain, including topological changes. This point is especially appreciated in the context of bifluid flows, where one may for instance want to track the motion of coalescing bubbles.

4. Solving the Level Set Hamilton-Jacobi Equation

We have seen in the previous sections that the motion of an domain \( \Omega(t) \) via a velocity field \( V(t,x) \) translates in terms of an associated level set function into the partial differential equations (2.6) or (2.7). Recall that \( V(t,x) \) may be of a quite intricate nature (in particular, depending on the evolving domain \( \Omega(t) \) in a complex way), which makes these equations difficult to solve in the general context.

As we have seen in Section 3, the numerical treatment of (2.6) or (2.7) most often requires to split the time interval \((0, T)\) into subintervals \((t^n, t^{n+1})\) of ‘small’ length \(\Delta t = t^{n+1} - t^n\), and to ‘freeze’ the velocity field on each subinterval:

\[
\forall t \in (t^n, t^{n+1}), \ V(t,x) \approx V(t^n, x).
\]

Doing so, the evolution equation (2.6) becomes a true advection equation on each interval \((t^n, t^{n+1})\), and can be solved relying on well-established numerical techniques (see e.g. [35]).

A slightly different point of view arises when one wants to preserve the information that the evolution of \( \Omega(t) \) is intrinsically oriented along the normal vector \( n_t(x) \), which is a crucial feature in several applications, such as shape optimization (see Chapter ??). Then, only the normal component of \( V \) is frozen:

\[
\forall t \in (t^n, t^{n+1}), \ V(t,x) \approx v(t^n, x)n_t(x),
\]

and the evolution equation (2.6) becomes a true Hamilton-Jacobi equation. However, the resolution of such equations is not so standard as that of advection equations.

In the present section, we discuss the numerical discretization of the Hamilton-Jacobi equation:

\[
\begin{aligned}
\frac{\partial \phi}{\partial t} + v|\nabla \phi| &= 0 \quad \text{on } (0, T) \times \mathbb{R}^d \\
\phi(0,.) &= \phi_0 \quad \text{on } \mathbb{R}^d
\end{aligned}
\]

for given normal velocity field \( v(x) \), and initial function \( \phi_0 \). The theory devoted to the study of numerical schemes for (4.1) is part of a more general theory associated to the numerical schemes for first order Hamilton-Jacobi equations of the form:

\[
\begin{aligned}
\frac{\partial \phi}{\partial t} + H(x, \nabla \phi) &= 0 \quad \text{on } (0, T) \times \mathbb{R}^d \\
\phi(0,.) &= \phi_0 \quad \text{on } \mathbb{R}^d
\end{aligned}
\]

and arises as the particular case when \( H(x,p) = v(x)|p| \). The forthcoming discussion takes place in this general context, which encompasses major applications outside from the level set framework, and perhaps allows to better appraise the salient features of the theory.

A wide variety of methods exists to deal with such problems, and we shall only sketch some basic ones.

For the sake of clarity, and without loss of generality, the forthcoming discussion holds in the two-dimensional case: \( d = 2 \).

4.1. Solving the Level Set Hamilton-Jacobi equation on Cartesian grids.

Notations: Let \( N \in \mathbb{N} \), and \( \Delta t = \frac{T}{N} \) be a time step, according to which \((0, T)\) is divided into subintervals \((t^n, t^{n+1})\), \( n = 0, ..., N - 1 \), with \( t^n = n\Delta t \). As for the space discretization, in this first subsection, the computational support is a Cartesian grid, with step \( \Delta x \) in the \( x \)-direction, and \( \Delta y \) in the \( y \)-direction. For a quantity \( \phi \) discretized at the vertices of the grid, and for \( i, j \in \mathbb{Z} \), \( \phi_{ij} \) denotes the value assigned to the node \( x_{ij} := (i\Delta x, j\Delta y) \).
Similarly to the case of hyperbolic systems of conservation laws, so-called upwind discretizations play a key role in the device of numerical schemes for Hamilton-Jacobi equations. Denote the finite difference quantities:

\[
D_{ij}^{+x}\phi = \frac{\phi_{i+1,j} - \phi_{ij}}{\Delta x} ; \quad D_{ij}^{-x}\phi = \frac{\phi_{ij} - \phi_{i-1,j}}{\Delta x},
\]

(and likewise for \(D_{ij}^{+y}\phi\) and \(D_{ij}^{-y}\phi\)), which are respectively referred to as forward and backward finite differences.

Our aim is to compute an approximation to the viscosity solution \(\phi\) of (4.2) as a sequence \(\phi^n = \{\phi^n_{ij}\}_{i,j \in \mathbb{Z}}\), with the meaning that \(\phi^n_{ij}\) is an approximation of \(\phi(t^n, x_{ij})\). An explicit, first-order numerical scheme which fulfills this role can be written under the general form:

\[
\begin{align*}
\forall i, j \in \mathbb{Z}, \quad & \phi^0_{ij} = \phi_0(i\Delta x, j\Delta y) \\
\forall n \in \mathbb{N}, i, j \in \mathbb{Z}, \quad & \phi^{n+1}_{ij} = \phi^n_{ij} - \Delta t \mathcal{H}(x_{ij}, D_{ij}^{-x}\phi^n, D_{ij}^{+x}\phi^n, D_{ij}^{-y}\phi^n, D_{ij}^{+y}\phi^n),
\end{align*}
\]

where the numerical Hamiltonian

\[
\mathcal{H}(x_{ij}, D_{ij}^{-x}\phi^n, D_{ij}^{+x}\phi^n, D_{ij}^{-y}\phi^n, D_{ij}^{+y}\phi^n)
\]

is intended as an approximation of \(H(x_{ij}, \nabla \phi(x_{ij}))\).

In the design of numerical schemes for (4.2), two properties prove desirable as for \(\mathcal{H}\):

**Definition 2.** An explicit, first-order scheme for (4.2) of the form (4.3) is said:

- consistent if, for any \(x \in \mathbb{R}^2\) and \(p \in \mathbb{R}^2\), \(\mathcal{H}(x, p_{x}, p_{y}, p) = H(x, p)\). In other words, the difference terms stand for the corresponding first-order derivatives where they should.
- monotone if, for any \(x \in \mathbb{R}^2\), and any \(i, j \in \mathbb{Z}\), the update function

\[
\{\phi_{ij}\}_{k, l \in \mathbb{Z}} \rightarrow \phi_{ij} - \Delta t \mathcal{H}(x, D_{ij}^{-x}\phi, D_{ij}^{+x}\phi, D_{ij}^{-y}\phi, D_{ij}^{+y}\phi)
\]

is increasing with respect to each of its arguments.

One can indeed prove that, under ‘reasonable’ assumptions over the theoretical Hamiltonian \(H\) and the initial data \(\phi_0\), consistent and monotone first-order schemes converge to the viscosity solution of (4.2) [20, 58].

As far as the particular Cauchy problem (4.1) is concerned, the simplest approximation consists in the following explicit first-order finite difference scheme:

\[
\begin{align*}
\forall n \in \mathbb{N}, i, j \in \mathbb{Z}, \quad & \phi^{n+1}_{ij} = \phi^n_{ij} - \Delta t \left(\max(v_{ij}, 0)\nabla_{ij}^+\phi^n + \min(v_{ij}, 0)\nabla_{ij}^-\phi^n\right) \\
\forall i, j \in \mathbb{Z}, \quad & \phi^0_{ij} = \phi_0(i\Delta x, j\Delta y)
\end{align*}
\]

where the discretizations \(\nabla_{ij}^+\phi\) and \(\nabla_{ij}^-\phi\) of \(|\nabla \phi|\) are defined as:

\[
\nabla_{ij}^+\phi = \left( \max(\max(D_{ij}^{-x}\phi, 0), -\min(D_{ij}^{+x}\phi, 0))^2 \right)^{\frac{1}{2}}
\]

and

\[
\nabla_{ij}^-\phi = \left( \max(\max(D_{ij}^{-y}\phi, 0), -\min(D_{ij}^{+y}\phi, 0))^2 \right)^{\frac{1}{2}}
\]

The quantity \(\nabla_{ij}^+\phi\) (resp. \(\nabla_{ij}^-\phi\)) is said to be upwind (resp. downwind), for its calculation only involves values of the \(\{\phi_{kl}\}_{k, l \in \mathbb{Z}}\) which are smaller (resp. larger) than \(\phi_{ij}\).

Accordingly, the discretization of the (exact) Hamiltonian \(H(x, p) = v(x)|p|\) by the numerical one:

\[
\forall n \in \mathbb{N}, i, j \in \mathbb{Z}, \quad H(x_{ij}, \nabla \phi(x_{ij})) \approx H_{ij}(\{\phi^n_{ij}\}_{k, l \in \mathbb{Z}}) := \max(v_{ij}, 0)\nabla_{ij}^+\phi^n + \min(v_{ij}, 0)\nabla_{ij}^-\phi^n
\]

can be deemed upwind in the sense that, for given \(i, j, n\), the update \(\phi^n_{ij} \rightarrow \phi^{n+1}_{ij}\) is only carried out using information coming from smaller values than \(\phi^n_{ij}\) if \(v_{ij}\) is positive, and larger values than \(\phi^n_{ij}\) if it is negative.
The numerical scheme (4.4) is consistent in the sense of Definition 2. However, it is not unconditionally monotone; it is indeed easily seen that a CFL-like relation connecting $\Delta t$ and $\Delta x, \Delta y$ must be satisfied for the latter property to hold. It reads:

$$\left( \sup_{i,j} v_{ij} \right) \frac{\Delta t}{\min(\Delta x, \Delta y)} \leq 1. \tag{4.7}$$

Grossly speaking, (4.7) means that the information should propagate over more than one space step $\Delta x$ or $\Delta y$ within one time step $\Delta t$.

Under this CFL condition, the scheme (4.4) turns out to be convergent. Moreover, an explicit error estimate of the discrepancy between the numerical solution $\{\phi_{kl}^n\}_{k,l \in \mathbb{Z}}$ and the exact viscosity solution $\phi(t,x)$ of (4.1) can be achieved:

$$\forall i,j \in \mathbb{Z}, \forall n \leq N, \ |\phi_{ij}^n - \phi(t^n, x_{ij})| \leq C\sqrt{\Delta t}, \tag{4.8}$$

for a constant $C$ depending only on the data of the problem (4.1) (the initial function $\phi_0$, etc.) We refer to [20, 58] for further details.

Unfortunately, this scheme is only first-order accurate, as is reflected by the ‘bad’ error estimate (4.8); in particular, it turns out to be very diffusive, meaning that sharp features of the exact solution $\phi$ to (4.1) tend to result smeared in the numerical solution $\{\phi_{kl}\}_{k,l \in \mathbb{Z}}$. It is thus desirable to devise higher-order schemes for solving (4.1,4.2). Mimicking conservation laws, it is possible to build high-order, adaptive-stencil schemes, known as (weighted) Essentially Non Oscillatory schemes (abridged as (w)ENO); see [30, 45], or the lecture notes [57] for further details.

4.2. Solving the Level Set Hamilton-Jacobi equation on triangular meshes.

The theory we just skimmed through in the previous subsection can be extended to the case of triangular meshes. Adequate generalizations of the notions of consistency and monotonicity introduced in Definition 2 make it possible to build a theory for convergent schemes on triangular meshes, a glimpse of which is now provided - see [1] for details.

**Notations:** As in the last subsection, the time interval $(0,T)$ is still split into subintervals $(t^n, t^{n+1})$, $n = 0, ..., N - 1$, $t^n = n\Delta t$.

The plane $\mathbb{R}^2$ is endowed with a conforming triangular mesh $\mathcal{T}$, and the vertices of $\mathcal{T}$ are denoted as $\{x_i\}_{i \in \mathcal{I}}$. If $\phi$ is a piecewise affine function on $\mathcal{T}$, $\phi_i$ is the value assigned to the node $x_i$. Recall that, in this context $\nabla \phi$ is a vector-valued function with constant values in restriction to each triangle of $\mathcal{T}$.

At each time $t^n$, an approximation of the viscosity solution $\phi(t^n, \cdot)$ to (4.2) is sought as a piecewise affine function $\phi^n$ on $\mathcal{T}$. A general explicit, first-order numerical scheme for (4.2) on mesh $\mathcal{T}$ can be written as:

$$\forall i \in \mathcal{I}, \quad \phi_i^{n+1} = \phi_i^n - \Delta t \mathcal{H}(x_i, \phi^n), \tag{4.9}$$

where $\mathcal{H}(x_i, \phi)$ is the numerical Hamiltonian. This notation may seem inappropriate at first glance, since the theoretical Hamiltonian $H$ only depends on $\phi$ through its gradient, and one could expect the same behavior from $\mathcal{H}(x_i, \phi)$. Actually, $\mathcal{H}(x_i, \phi)$ will depend only on $\nabla \phi$, but it is better expressed in terms of $\phi$.

Let us now outline one possible procedure for constructing the numerical Hamiltonian $\mathcal{H}(x_i, \phi)$, which is very reminiscent of Lax-Friedrichs numerical schemes in the context of hyperbolic systems of conservation laws. Let $C(H)$ be one Lipschitz constant for the theoretical Hamiltonian $H$; if $h > 0$ is smaller than the smallest edge of $\mathcal{T}$, one defines:

$$\forall i \in \mathcal{I}, \quad \mathcal{H}(x_i, \phi) = H(x_i, \overline{\phi}_i) - \frac{C(H)}{2\pi h} \int_{D_h(x_i)} (\phi(x) - \phi(x_i)) \, d\ell(x),$$

where $\overline{\phi}_i$ is the mean value of $\nabla \phi$ over the disk $D_h(x_i)$ of center $x_i$ and radius $h$ (see figure 9):

$$\overline{\phi}_i = \frac{1}{\pi h^2} \int_{D_h(x_i)} \nabla \phi \, dx,$$
and $C_h(x_i) := \partial D_h(x_i)$ is the circle of center $x_i$ and radius $h$.

\[ \partial D_h(x_i) \]

**Figure 9.** The numerical scheme for triangular meshes of Section 4.2: setting and notations.

One can show that the sequence $\{\phi^n\}_{n=0,\ldots,N}$ obtained using (4.9) converges (in an appropriate sense) to the viscosity solution of (4.2), provided the following CFL condition holds:

\[ \Delta t \leq \frac{h}{C(H)}. \]  

**Remark 2.** Other methods exist for solving (4.1,4.2) on a triangular mesh; see for instance [10], where, in particular, so-called Petrov-Galerkin approximations are introduced, featuring a usual Finite Element formulation which is stabilized by the addition of least-square quantities.

### 4.3. Semi-Lagrangian schemes.

So-called semi-Lagrangian techniques attempt to compute the solution $\phi$ to (4.1) or (4.2) by tracking the flow of information attached to the equation. In this section, we illustrate the main underlying idea with a short and formal description of the semi-Lagrangian scheme of Strain [59] for the level set Hamilton-Jacobi equation (4.1): as, in their broad lines, semi-Lagrangian methods are fairly independent of the particular choice of a computational support, we purposely do not specify which space discretization is used for functions.

Let us as usual subdivide the time interval $(0, T)$ into subintervals of the form $(t^n, t^{n+1})$. The essence of semi-Lagrangian schemes is to compute the value $\phi(t^{n+1}, x)$ of the solution $\phi$ at time $t^{n+1}$ and at a (grid) point $x \in \mathbb{R}^d$ by tracking the point $y \in \mathbb{R}^d$ such that the information of $\phi(t^{n+1}, x)$ ‘comes from’ $\phi(t^n, y)$. To be more precise, first recall that equation (4.1) stems (at least in a formal way) from the level set advection equation:

\[ \partial \phi \partial t + V(t,x) \cdot \nabla \phi = 0 \quad \text{on } (0, T) \times \mathbb{R}^d \]

\[ \phi(0,.) = \phi_0 \quad \text{on } \mathbb{R}^d, \]

where the vector velocity field $V : (t,x) \mapsto V(t,x) \in \mathbb{R}^d$ depends on $\phi$ and is defined as:

\[ V(t,x) = v(t,x) \frac{\nabla \phi(t,x)}{||\nabla \phi(t,x)||}. \]

The nonlinear equation (4.11) is approximated on each subinterval $(t^n, t^{n+1})$ by the linear advection equation obtained by freezing the value of $V(t,.)$ over $(t^n, t^{n+1})$, i.e. by setting:

\[ \forall t \in (t^n, t^{n+1}), \forall x \in \mathbb{R}^d, \quad V(t,x) = V(t^n,x) =: V^n(x). \]

In other terms, assuming that an approximation $\phi^n$ of $\phi(t^n,.)$ has already been computed, $\phi(t,.)$ is sought on $(t^n, t^{n+1})$ as the solution $\psi : (t^n, t^{n+1}) \times \mathbb{R}^d \to \mathbb{R}$ to:

\[ \partial \psi \partial t + V^n(x) \cdot \nabla \psi = 0 \quad \text{on } (0, T) \times \mathbb{R}^d \]

\[ \psi(t^n,.) = \phi^n \quad \text{on } \mathbb{R}^d, \]
whose exact solution can be computed owing to the *method of characteristics* (see Chapter ?? for a similar use in the context in fluid mechanics):

\[
(4.12) \quad \forall x \in \mathbb{R}^d, \quad \psi(t^{n+1}, x) = \psi(t^n, \chi(x, t^n, t^{n+1})) = \phi^n(\chi(x, t^n, t^{n+1})),
\]

where \((t^n, t^{n+1}) \ni t \mapsto \chi(x, t, t^{n+1})\) is the *characteristic curve* of \(V^n\), reaching \(x\) at time \(t^{n+1}\), defined by (2.5).

With this guidance in hand, the following procedure for approximating the solution \(\phi\) to (4.1) is derived:

- **Initialization:**
  Start with an approximation \(\phi^0\) of \(\phi(0,.)\) on the considered computational support (e.g. at the nodes of a Cartesian grid, or as a piecewise linear function on a simplicial mesh).

- **Loop (for \(n = 1,...,N - 1\)):**
  **Loop (for each degree of freedom \(x\) of the computational support):**
  
  (1) Calculate \(t \mapsto \chi(x, t, t^{n+1})\), and search for the ‘foot’ \(y := \chi(x, t^n, t^{n+1})\) of the characteristic curve reaching \(x\) at time \(t^{n+1}\) (see Chapter ?? for numerical methods to achieve this).

  (2) Mimicking formula (4.12), the value of \(\phi^n\) at \(y\) is computed (using interpolation on the computational support) to produce \(\phi^{n+1}(x)\).

The benefits of this approach are numerous: among others, it is easily parallelized since the nodes of the mesh are consistently processed independently from one another. What’s more, the stability of the method does not depend on a CFL condition over the time step \(\Delta t\) such as (4.7,4.10). On the other hand, this technique also has some drawbacks, such as the lack of accuracy in regions where the exact solution \(\phi\) is not smooth.

This semi-Lagrangian paradigm (which in the hitherto considered case of Equation (4.1) boils down to solving (2.6)) may be extended to general Hamilton-Jacobi equations of the form (4.2); the particular situation discussed above exemplifies the three typical stages of a general semi-Lagrangian method. Between two consecutive time steps \(t^n, t^{n+1}\), an approximation \(\phi^{n+1}(x)\) of the value \(\phi(t^{n+1}, x)\) is computed using:

- **(1)** A *space-time integration step*, which amounts to searching for the point \(y \in \mathbb{R}^d\) such that ‘the information about \(\phi(t^{n+1}, x)\) comes from \(\phi(t^n, y)\)’ (in our case, \(y\) is the foot \(\chi(x, t^n, t^{n+1})\) of the characteristic curve passing through \(x\) at time \(t^{n+1}\)).

- **(2)** A *spatial interpolation step*, during which \(\phi^n(y)\) is interpolated from the values of \(\phi^n\) at the grid nodes surrounding \(y\).

- **(3)** An *update step*, during which \(\phi^{n+1}(x)\) is computed from \(\phi^n(y)\) (in the presented case, this step is trivial, thanks to formula (4.12)).

Using this idea in the context of a Hamilton-Jacobi equation of the form (4.2) demands a means to complete the first and third stages of the previous program. This is generally achieved thanks to representation formulae for the exact solutions to (4.2), which generalize the method of characteristics in the case of more general Hamiltonian functions \(H\) (e.g. the Hopf-Lax formula when \(H\) is convex and does not depend on the \(x\) variable). See [25] for more details.

5. Initializing level set functions

We have already pointed out the fact that many level set functions can be associated to a domain \(\Omega \subset \mathbb{R}^d\).

The theoretical framework of Section 2 is independent of which function \(\phi\) is chosen, as long as it fulfills (2.1). Things are very different in numerical practice. It turns out that too steep or too loose variations of \(\phi\) near \(\Gamma\) may cause instabilities in locating accurately \(\partial \Omega\), or difficulties evaluating precisely the normal vector or curvatures of \(\Gamma\) by means of formulae such as (2.2,2.3,2.4) (see e.g. [15]). This pleads for initializing a level set function for \(\Omega\) as the distance function - and more precisely (for sign purposes) as the *signed distance function* \(d_\Omega\) to \(\Omega\), that is:

\[
\forall x \in \mathbb{R}^d, \quad d_\Omega(x) = \begin{cases} 
-d(x, \partial \Omega) & \text{if } x \in \Omega \\
0 & \text{if } x \in \Gamma \\
d(x, \partial \Omega) & \text{if } x \in \partial \Omega
\end{cases}
\]
Indeed, we have seen in Example 1 that this function features a very ‘regular’ spacing out of its isolines. Besides, among its appealing properties, it turns out that \( d_\Omega \) is smooth near \( \partial \Omega \), provided \( \Gamma \) is a smooth boundary.

In this section, we describe several ‘classical’ algorithms for generating the (signed or unsigned) distance function to a domain \( \Omega \) on a computational support (see also Section 7.1). Most numerical methods for generating the signed distance function to a domain \( \Omega \) are more generally devoted to Eikonal equations of the form (2.9) or (2.10). The Fast Marching Method, originally introduced in [53], is no exception in this regard.

5.1. The Fast Marching Method on Cartesian grids.

Let us briefly present the Fast Marching Method for generating the (unsigned) distance function to a bounded domain \( \Omega \subset \mathbb{R}^2 \), at those nodes of a Cartesian grid of the plane lying outside \( \Omega \) (i.e. we solve (2.9)). Extensions of this discussion to the case of nodes lying inside \( \Omega \) and to the three-dimensional setting are rather straightforward.

Reusing the notations of section 4.1, the fast marching method is an iterative process which produces at each step \( n \in \mathbb{N} \) a scalar numerical quantity \( T := (T^n_{ij})_{i,j \in \mathbb{Z}} \) intended as an increasingly accurate approximation of the Euclidean distance \( d(., \Gamma) \).

It relies first on an update strategy, according to which the values of \( T \) are computed in a one-by-one, upwind fashion, mimicking the propagation of a front starting from \( \Gamma \). More accurately, the nodes \( x_{ij} \) of the Cartesian grid are consistently parted into three categories:

- The accepted nodes: those are the nodes \( x_{ij} \) ‘where the front has already passed’, i.e. at which the current value \( T^n_{ij} \) is assumed converged. Once a value has become accepted, it is no longer updated.
- The active nodes: those are the nodes \( x_{ij} \) ‘on the front’. One of their four neighbors \( x_{i-1j}, x_{i+1j}, x_{ij-1}, x_{ij+1} \) or \( x_{ij+1} \) is an accepted node, and a first approximation (trial value) \( T^n_{ij} \) to the desired solution has been computed, but may still be subject to updates.
- The far nodes: those are the nodes ‘still far from the front’, whose values have not even been approximated (and are set to \( \infty \)).

At each iteration, the algorithms accepts one node, to be selected among the set of active nodes. The set of active nodes is then redefined, and their values are updated according to this new information about the front.

The update procedure \( T^n_{ij} \rightarrow T^{n+1}_{ij} \) of the value of \( T^n \) at an active node \( x_{ij} \) is the second key feature of the algorithm. At every such node, a trial value \( \tilde{T}^n_{ij} \) is computed as the solution to the following equation, which is obtained as an upwind discretization of the Eikonal equation (2.9):

\[
\min \left( \frac{T^n_{i+1j} - T^n_{ij}}{\Delta x}, 0 \right), \min \left( \frac{T^n_{i-1j} - T^n_{ij}}{\Delta x}, 0 \right) \right)^2 + \min \left( \frac{T^n_{ij+1} - T^n_{ij}}{\Delta y}, 0 \right), \min \left( \frac{T^n_{ij-1} - T^n_{ij}}{\Delta y}, 0 \right) \right)^2 = \frac{1}{c_{ij}}.
\]

Note that this rule is intrinsically upwind, since the derived value \( \tilde{T}^n_{ij} \) is only influenced by the values of \( T^n \) at the four neighbors of \( x_{ij} \) that are smaller than \( \tilde{T}^n_{ij} \): this is a means to impose a causality principle which is inherent to Hamilton-Jacobi equations. Only the accepted values among the set \( \{T^n_{i-1j}, T^n_{i+1j}, T^n_{ij-1}, T^n_{ij+1}\} \) are used in the second order polynomial equation (5.1), and it must be checked that the obtained solution \( \tilde{T}^n_{ij} \) is larger than those values. In the end, \( T^{n+1}_{ij} \) is obtained as:

\[
T^{n+1}_{ij} = \min \left( \tilde{T}^n_{ij}, T^n_{ij} \right).
\]

To sum up, the Fast Marching algorithm proceeds along the lines of the following sketch:

- Initialization:
Figure 10. Setting of the Fast Marching Method

(1) Compute the exact distance function at the nodes of the cells which intersect $\Gamma$, and mark them as accepted.
(2) Use the local update procedure (5.1) to compute a trial value at the neighbor points to the accepted points which have not been yet accepted, and mark them as active.
(3) Mark all the remaining nodes as far, and assign them value $\infty$.

• **Loop (while the set of active nodes is non empty):**
  (1) Travel the set of active nodes, and identify the one with minimum trial value. This node becomes accepted.
  (2) Identify the new set of active nodes, and compute a new trial value for each one of them, using the local update solver (5.1) for the Eikonal equation.

This algorithm produces a sequence $(T^n_{ij})$ which converges towards the unique viscosity solution to (2.9); see [21] for a precise statement of this fact and a proof.

Furthermore, it can be seen that, if in practice, the computation is restrained to a large bounding domain (e.g. a box), equipped with a Cartesian mesh consisting of $N$ vertices, the Fast Marching procedure converges within $O(N \log(N))$ operations.

5.2. Extension of the Fast Marching Method to triangular meshes.

Let us now give a hint of how the Fast Marching Method can be adapted to the context of a triangular mesh of $\mathbb{R}^2$, using the notations introduced in section 4.2, and following the work [34].

The general outline of the previous algorithm is unchanged: the vertices of the mesh are still tagged as either accepted, active, or far. At each iteration, the active vertex whose value is minimal becomes accepted once and for all. The set of active vertices is then adequately redefined, and the active values are updated.

The update procedure is actually the only very different feature between both versions of the Fast Marching algorithm. Here, it occurs in a situation where two values of $T$, say $T_i \leq T_j$ are known at two vertices $x_i, x_j$ of a triangle $K = x_i x_j x_k \in \mathcal{T}$, and a trial value $\tilde{T}_k$ is sought at $x_k$.

To this end, $T$ is approximated by its piecewise linear interpolate $\pi_K T$ on $T$ from the accepted values $T_i, T_j$ and the sought trial value $\tilde{T}_k$ at $x_i, x_j, x_k$ respectively. Let $c_K$ be an approximation of $c$ over $K$ (e.g. $c_K$ may be the mean value of $c$ over $K$). $\tilde{T}_k$ should be such that:

$$|\nabla (\pi_K T)|^2 = c_K^2.$$
\( \tilde{T}_k \) is then searched as a solution to this quadratic equation which is larger than \( T_i \) and \( T_j \), and satisfies some additional properties (related to the causality inherent to equation 2.9 discussed above), which are omitted here. If such a solution \( \tilde{T}_k \) exists, the value \( T_k \) is updated as \( T_k = \min(T_k, \tilde{T}_k) \).

Let us mention a potential difficulty in this approach. Depending on the shape of the triangulation \( \mathcal{T} \), it may happen that, in the course of the update of the value \( T_k \) at point \( x_k \), no triangle having \( x_k \) as a vertex has accepted values at both other vertices, which makes it impossible to rely on the previous local procedure to compute a new trial value \( \tilde{T}_k \) at \( x_k \). This is especially likely to happen when the angles of triangles at \( p_k \) are obtuse. A special procedure is required in this case to reduce to the previous case.

The exact same construction can be used to generate the distance function to a subset on a triangulated surface of \( \mathbb{R}^3 \) (this is actually the original setting of the work [34]). Yet, this procedure is more difficult to extend to the case of a tetrahedral computational mesh of \( \mathbb{R}^3 \).

**Remark 3.** Again, many other numerical methods exist to solve Eikonal equations of the form (2.9); for an alternative technique, the so-called Fast Sweeping Method, see [66].

### 6. Operations within the Level Set framework

The Level Set Method features an implicit description of the domain \( \Omega \) under consideration; however well-suited this framework for tracking its evolution, as we have extensively discussed, the absence of any explicit discretization (e.g. a mesh) of \( \Omega \) may be the cause of difficulties when performing certain operations.

We hereafter list the most common ones, as well as popular numerical recipes to achieve them.

Throughout this section, \( \Omega \) stands for a (smooth) bounded domain in \( \mathbb{R}^d \) with boundary \( \Gamma \), and \( \phi \) is an associated Level Set function.

#### 6.1. Evaluation of the normal vector or the mean curvature of \( \Gamma \).

Relying on the theoretical expression (2.2) of the normal vector \( n \) to \( \Gamma \) (pointing outward \( \Omega \)), a numerical approximation can be achieved by the following formula:

\[
n(x) \approx \frac{\nabla \phi(x)}{\sqrt{\|
abla \phi(x)\|^2 + \varepsilon^2}},
\]

in which \( \varepsilon \) is a ‘very small’ parameter (typically of the order of \( 10^{-5} \)) acting as a safeguard against degeneracy of \( \nabla \phi \). The way to evaluate numerically the involved partial derivatives depends on the computational support at hand; for instance, when \( \phi \) is discretized at the vertices of a Cartesian grid, finite difference approximations of various orders may be used, for example, simple first-order finite differences or more sophisticated ENO approximations.

Likewise, an approximation of the mean curvature \( \kappa(x) \) of \( \Gamma \) is based on Formula (2.4):

\[
\kappa(x) \approx \text{div} \left( \frac{\nabla \phi(x)}{\sqrt{\|
abla \phi(x)\|^2 + \varepsilon^2}} \right).
\]

Note that both formulae actually make sense outside from \( \Gamma \), so that they actually are approximations to extended normal vector and mean curvature fields.

#### 6.2. Calculating integrals on \( \Omega \) and \( \Gamma \).

Let us now address the numerical evaluation of volume or surface integrals of the form:

\[
I = \int_{\Omega} f(x) \, dx, \quad \text{and} \quad J = \int_{\partial \Omega} g(x) \, ds,
\]

where \( f, g: \mathbb{R}^d \to \mathbb{R}^d \) are given (smooth) functions. A simple calculation of \( I \) relies on the following approximation of the characteristic function \( \chi_{\Omega} \) of \( \Omega \) in terms of \( \phi \):

\[
\chi_{\Omega}(x) \approx H_{\varepsilon}(\phi(x)) := \frac{1}{2} \left( 1 - \frac{\phi(x)}{\sqrt{\phi^2(x) + \varepsilon^2}} \right), \quad x \in \mathbb{R}^d
\]

[19]
where $H_\varepsilon$ is a smoothed approximation of the characteristic function of the interval $(-\infty, 0)$ in $\mathbb{R}$, and $\varepsilon$ is again a ‘small parameter’. In turn, $I$ can be approximated as:

$$I \approx \int_{\mathbb{R}^d} f(x) H_\varepsilon(\phi(x)) \, dx,$$

the last integral being straightforward to evaluate relying on appropriate quadrature formulae, and depending on the computational support where $\phi$ is discretized.

The approximation of the integral $J$ is slightly more intricated; it relies on an approximation of the surface measure of $\Gamma$, that is, of the distribution $\delta_\Gamma \in D'(\mathbb{R}^d)$ defined as the integration over $\Gamma$:

$$\forall \varphi \in C_c^\infty(\mathbb{R}^d), \quad \langle \delta_\Gamma, \varphi \rangle = \int_\Gamma \varphi \, ds.$$

An application of Green’s formula (see Proposition ?? in Chapter ??) reveals that the following identity holds, in the sense of distributions:

$$\delta_\Gamma = -\frac{\partial \chi_\Omega}{\partial n},$$

where $n$ is a divergence-free extension of the normal vector to $\Gamma$,

whence we base the approximation:

$$\delta_\Gamma \approx -\frac{\partial}{\partial n} (H_\varepsilon(\phi)).$$

Then, the following formula lends itself to an easy numerical evaluation:

$$J \approx -\int_{\mathbb{R}^d} \frac{\partial}{\partial n} (H_\varepsilon(\phi(x))) g(x) \, ds.$$

### 6.3. Algebraic operations over sets.

The main algebraic operations over domains $\Omega, \Omega_1, \Omega_2 \subset \mathbb{R}^d$ may be achieved by simple manipulations over corresponding level set functions $\phi, \phi_1, \phi_2$:

- A level set function $\phi_c$ for the complement $\complement \Omega$ of $\Omega$ is obtained as:

  $$\phi_c = -\phi.$$

- A level set function $\phi_u$ for the union $\Omega_1 \cup \Omega_2$ is:

  $$\phi_u = \min(\phi_1, \phi_2).$$

- A level set function $\phi_i$ for the intersection $\Omega_1 \cap \Omega_2$ is:

  $$\phi_i = \max(\phi_1, \phi_2).$$

Notice that even if $\phi_1$ and $\phi_2$ are smooth, $\phi_u$ and $\phi_i$ may fail to be so (as the corresponding domains do).

### 6.4. Solving partial differential operations posed on $\Omega$.

Let $\Omega$ be a domain of interest, included in a larger, bounded domain $D \subset \mathbb{R}^d$. In the practical setting, $D$ is the computational domain, and comes with a mesh (a Cartesian grid, a triangular mesh, etc.); $\Omega$ is solely known via an associated level set function $\phi$, which is discretized on the mesh of $D$; in particular, no mesh of $\Omega$ is available. Let us consider the resolution of the following partial differential equation on $\Omega$:

\begin{equation}
\begin{cases}
-\text{div}(a \nabla u) + u = f & \text{in } \Omega, \\
\frac{\partial u}{\partial n} = 0 & \text{on } \Gamma,
\end{cases}
\end{equation}

where $a > 0$ and $f \in L^2(\mathbb{R}^d)$. In the present situation, it seems difficult to rely on classical numerical methods (e.g. with the Finite Element method) since no mesh of $\Omega$ is available.

The idea is to approximate the solution $u$ to (6.2) with that $u_\varepsilon$ to an approximate problem, defined on the whole working domain $D$, in such a way that:

$$u_\varepsilon \approx u \text{ on } \Omega.$$
Multiple ways exist to achieve this purpose. One possibility is to consider the following approximate problem:

\begin{align}
\begin{cases}
-\text{div}(c_\varepsilon(x)a\nabla u) + c_\varepsilon(x)u = c_\varepsilon(x)f & \text{in } D, \\
u = 0 & \text{on } \partial D, \\
\end{cases}
\end{align}

(6.3)

where \(D\) is the void region \(D \setminus \Omega\) has been filled with a very soft material. It is indeed possible to prove that:

\[ ||u - u_\varepsilon||_{H^1(\Omega)} \xrightarrow{\varepsilon \to 0} 0.\]

This problem (6.3) proves now easier to simulate using the data at hand than (6.2): it is posed on the (meshed) domain \(D\), and the cutoff \(c_\varepsilon(x) = \varepsilon + \chi_\Omega(x)(1-\varepsilon)\) can be approximated as:

\[ c_\varepsilon(x) \approx \varepsilon + H_\varepsilon(\phi(x))(1 - \varepsilon), \]

where \(H_\varepsilon(\phi)\) is the approximate characteristic function of \(\Omega\) based on the level set function \(\phi\) (6.1).

**Remark 4.**

- The approximate equation (6.3) intrinsically depends on the exact equation (6.2) to be solved. For instance, if one considers the problem

\[
\begin{cases}
-\text{div}(\nabla u) + u = f & \text{in } \Omega, \\
u = 0 & \text{on } \partial \Omega,
\end{cases}
\]

instead of (6.2), one may show that \(u\) has to be approximated with the solution \(u_\varepsilon\) of:

\[
\begin{cases}
-\text{div}(\nabla u) + c_\varepsilon(x)u = f & \text{in } D, \\
u = 0 & \text{on } \partial D, \\
\end{cases}
\]

where \(c_\varepsilon(x) = \begin{cases} 1 & \text{for } x \in \Omega, \\ \frac{1}{\varepsilon} & \text{for } x \in D \setminus \Omega. \end{cases}\)

and no longer with that of (6.3).

- Similar approximation processes hold, e.g. for the linear elasticity equations (this is the so-called *Ersatz material approach*), or for Stokes equations (see Chapter ??, §??).

7. **Additional features**

We now describe several additional issues that regularly come up in the numerical implementation of the Level Set Method. In this section, \(\Omega(t)\) stands for a domain evolving a period of time \((0, T)\), and \(\phi(t, \cdot)\) is an associated level set function. To set ideas, we focus on the situation where the motion is driven by a normal velocity given by the scalar function \(v(t, x)\); consequently, \(\phi\) arises as the solution to (4.1).

### 7.1. Redistancing.

We already pointed out the practical importance of manipulating level set functions that ‘resemble’ signed distance functions. Unfortunately, even if \(\phi(0, \cdot)\) is the signed distance function to \(\Omega(0)\), the solution \(\phi(t, \cdot)\) to the level set equation (4.1) is likely to develop very sharp or loose variations in the vicinity of \(\Gamma(t)\) as \(t > 0\) increases. This phenomenon may jeopardize the stability of the whole numerical resolution of (4.1).

In practical applications, it turns out crucial to periodically restore \(\phi(t, \cdot)\) as the signed distance function to \(\Omega(t)\): this is the purpose of *level set redistancing* (or re-initialization).

The situation is as follows: \(\Omega \subset \mathbb{R}^d\) is a bounded domain, about which a (possibly very ‘irregular’) level set function \(\phi_0\) is available; we aim at generating the signed distance function \(d_\Omega\) to \(\Omega\) from these data. The most straightforward idea in this direction consists of course in using one of the algorithms of Section 5. Doing so would be a waste somehow, since we already have one level set function for \(\Omega\), and we could hope to take advantage of this knowledge. In such a situation, it has been proposed in [60] to ‘regularize’ the level set function \(\phi_0\) into a new one, close to \(d_\Omega\). To this end, \(\phi_0\) is used as the initialization of the *redistancing equation*:

\[
\begin{cases}
\frac{\partial \psi}{\partial t}(t, x) + \text{sgn}(\phi_0(x))(|\nabla \psi| - 1) = 0 & \text{for } (t, x) \in (0, \infty) \times \mathbb{R}^d \\
\psi(0, x) = \phi_0(x) & \text{for } x \in \mathbb{R}^d.
\end{cases}
\]

(7.1)

The underlying intuition is that, as the stationary state of (7.1) is obtained, the property \(|\nabla \psi| = 1\) is restored (which is formally obtained in the above equation by cancelling the time derivative). The presence of the
sign function accounts for the fact that a signed distance function is sought. This very formal explanation is theoretically justified in [6].

In practice, the redistancing equation (7.1) is solved using adequate numerical schemes, in the spirit of those presented in Section 4 and 5, relying on a smoothing of the discontinuous function \( \text{sgn}(\phi_0) \).

**Remark 5.** However crucial, this redistancing procedure comes with unwelcome side effects; first, using it ‘too often’ may undermine the efficiency of the process in terms of CPU time (however, in most applications, the cost of this procedure is nothing to be compared with, e.g., that of a finite element calculation). A more serious issue is that redistancing ‘too often’ may entail a loss of accuracy of the process, since the interface \( \Gamma \) is slightly shifted in the process of solving (7.1) (that is, the 0 level set of \( \psi(t,\cdot) \) does not match exactly that of \( \phi_0 \)). Therefore, a balance has to be reached between redistancing ‘often’, and not ‘too often’.

### 7.2. Velocity extension.

Another recurring difficulty in the practice of the Level Set Method is that, in a wide class of applications, the (scalar) velocity field \( v(t,x) \) is only defined at those points \( x \) lying on the boundary \( \Gamma(t) \) of the evolving domain. This is a pity, for as we saw in Section 2.2, the Level Set Method requires it to be defined over \( \Omega(t) \), and we briefly outline two popular ones in the literature, with different assets.

#### 7.2.1. Normal extension of the velocity.

In [4], the authors propose a velocity extension approach which alleviates the need for redistancing the level set function. This relies on the following observation: assume that, for \( t > 0 \), the solution \( \phi(t,\cdot) \) to the level set Hamilton-Jacobi equation (4.1) with normal velocity \( v \) is the signed distance function to \( \Omega(t) \); a formal calculation, blithely ignoring regularity issues reveals, in this context:

\[
0 = \frac{\partial}{\partial t} (|\nabla \phi|^2) = -2|\nabla \phi|\nabla \phi \cdot \nabla v - 2v|\nabla \phi| \nabla (|\nabla \phi|) = -2\nabla \phi \cdot \nabla v \quad \text{on } \mathbb{R}^d,
\]

where we used Equation (4.1) and the property \( |\nabla \phi| = 1 \). Hence, if we are to select the extended velocity field \( v_{\text{ext}} \) so that \( \phi(t,\cdot) \) stays a signed distance function, it must obey the equation (see the appendix in [67] for further discussion):

\[
\nabla v_{\text{ext}}(t,x) \cdot \nabla \phi(t,x) = 0. \tag{7.2}
\]

Recalling Formula (2.2) for the (extended) normal vector \( n_t \) to \( \Gamma(t) \), (7.2) means that, at every time \( t \), \( v_{\text{ext}}(t,\cdot) \) should be constant along \( n_t \).

Let us now slip into the numerical setting: recall that the time interval \((0,T)\) is divided into subintervals \((t^n,t^{n+1})\) of length \( \Delta t \), \( n = 0, ..., N = T/\Delta t \). The Hamilton-Jacobi equation (4.1) is solved on each such interval \((t^n,t^{n+1})\) with initial data \( \phi^n := \phi(t^n,\cdot) \), and velocity field \( v^n := v(t^n,\cdot) \). This velocity field is defined on \( \Gamma(t^n) \) and has to be extended to, say, \( \mathbb{R}^d \).

One method to stick with the previous observations - insofar as possible - goes the following way:

1. Calculate the signed distance function \( d_{\Omega^n} \) to \( \Omega^n \). This can be achieved thanks to one of the methods of Section 5, or, more efficiently, by solving the redistancing equation (7.1) (since we have one level set function \( \phi^n \) for \( \Omega^n \) at hand).
2. Calculate \( v_{\text{ext}} \) as the solution to:

\[
\begin{cases}
\nabla v_{\text{ext}} \cdot \nabla d_{\Omega^n} = 0 & \text{in } \mathbb{R}^d \setminus \Gamma(t^n) \\
v_{\text{ext}} = v^n & \text{on } \Gamma(t^n).
\end{cases}
\]

This system happens to share much features with boundary-value problems of the form (2.10) and can be solved owing to a very similar procedure as the Fast Marching Method.
We refer to the article [16] for further discussion on this method, and for a similar procedure in the context of Fast Marching Methods.

### 7.2.2. PDE-based extension.

A different extension technique allows to extend the velocity field \( v \) into a field \( v_{\text{ext}} \) which is guaranteed to be smooth. Let us describe this procedure at a fixed iteration of a Level Set evolution procedure: we have a domain \( \Omega \) (standing for \( \Omega^n \) at the considered time \( t^n \)), and wish to extend \( v \equiv v^n \) to the larger computational domain \( D \).

One possibility is to search for the unique solution \( v_{\text{ext}} \in H^1(D) \) to the following equation:

\[
\begin{cases}
-\alpha \Delta v_{\text{ext}} + v_{\text{ext}} = 0 & \text{in } D, \\
v_{\text{ext}} = v & \text{on } \partial \Omega.
\end{cases}
\]

Doing so guarantees some degree of smoothness, since \( v_{\text{ext}} \) intrinsically belongs to \( H^1(D) \).

This procedure is more expensive than that described in the previous section, since it involves the resolution of the elliptic equation (7.3) (which is typically achieved thanks to a finite element calculation). However, it admits other, very interesting applications, for instance if we relax the constraint that \( v_{\text{ext}} = v \) on \( \Gamma \); taking such freedom may be unacceptable in some applications (for instance that of Section 3), but may even be very desirable in some other, for instance in shape optimization. We refer to Chapter ??, Section ??, where this issue is further discussed in this context.

### 7.3. Towards enhancing the efficiency of the Level Set method: the Narrow Band approach.

At this point, it is worth wondering about the efficiency of the Level Set method. Indeed, the problem of evolution of a surface is traded for that of a quantity defined on a space of a higher dimension, and one may expect that this entails a significant increase in computational cost.

Several techniques have been thought up to increase this efficiency, among which parallel implementations [36], and mesh adaptation procedures (see e.g. [10], or the recent [2]). However interesting, we do not get into the details of these methods and refer to the aforementioned articles.

Another key improvement of the level set method is the so-called Narrow Band approach. In most applications where the evolution of a domain \( \Omega(t) \) (or that of its boundary \( \Gamma(t) \)) is described by that of an associated Level Set function \( \phi(t, \cdot) \), only the motion of the 0 level set of \( \phi \) is of interest. As was originally remarked in [15], then systematized in [3], it is thus enough to carry out the operations related to the Level Set Method (initializing the level set function, redistancing it, solving Equation (4.1), extending the velocity field) only on a narrow band around \( \Gamma(t) \). Here is a very rough sketch of this paradigm, in which we retain the Cartesian setting and the notations of Section 4.1 (see Figure 11 for an illustration).

**Loop (From \( n = 0 \), while \( n < N \)):**

- **Initialization** of a narrow band \( B \) of ‘Near points’ around the actual interface \( \Gamma(t^n) \). It may be constructed by relying on the distance function to \( \Gamma(t^n) \), or simply as a tube of \( k \) elements around \( \Gamma(t^n) \).
- **Loop:**
  1. Perform an attempt iteration
     
     \[
     \forall i, j \in \mathbb{Z} \text{ s.t. } x_{ij} \in B, \quad \{ \phi_{ij}^n \} \mapsto \{ \phi_{ij}^{n+1, \text{temp}} \}
     \]
     
     in the solution of the Level Set Hamilton-Jacobi equation (4.1) using, e.g., the numerical scheme of Section 4.1. Only the nodes of the grid lying inside the narrow band are updated, and a special attention must be paid to the calculation of derivatives at the points of \( B \) close to its border. The process can be even more sped up by using adapted data structures for the storage of the nodes in the narrow band.
  2. Appraise the position of the attempt front \( \Gamma_{\text{temp}}(t^{n+1}) \) (this can be done by looking at the sign of the quantity \( \phi_{\text{temp}}^{n+1} \) at the nodes lying on the border of the narrow band):
– If $\Gamma_{\text{temp}}(t^{n+1})$ lies inside the narrow band $B$, accept the iteration:

$$n \leftarrow n + 1, \quad \{\phi_{ij}^n\} \leftarrow \{\phi_{ij,\text{temp}}^{n+1}\},$$

and go back to (1).

– Else, refuse the iteration, interrupt the loop, and go back to the Initialization step.

![Figure 11. Setting of the Narrow Band Method; a band $B$ (the greyed region) of close points (in black) is constructed around the interface $\Gamma(t^n)$ (in red), and the other points of the grid are tagged as far. Updates $\phi_{ij}^n \rightarrow \phi_{ij}^{n+1}$ are performed only for black points $x_{ij}$ as long as the front $\Gamma(t^n)$ stays inside $B$.](image)

### 7.4. Handling multiple phases with the Level Set method.

We have hitherto been using the Level Set method to account for the motion of one domain, that we shall denote as $\mathcal{O}(t)$ in this section - and only in this section. Equivalently, this amounts to describing the motions of the two phases $\Omega_0 := \mathcal{O}$ and $\Omega_1 := D \setminus \overline{\mathcal{O}}$, where $D$ is the fixed computational domain, and where we dropped the mention to time. Note that this point of view was already used in Section 3.

As was noticed in [63], a small increment in this method allows to describe multiple phases: using $m$ domains $\mathcal{O}_0, \ldots, \mathcal{O}_{m-1} \subset D$ it is possible to represent up to $2^m$ domains $\Omega_0, \ldots, \Omega_{2^m-1}$, obtained as combinations of the $\mathcal{O}_i$, (see Figure 12):

\[
\begin{align*}
\Omega_0 &= \mathcal{O}_0 \cap \mathcal{O}_1 \cap \ldots \cap \mathcal{O}_{m-2} \cap \mathcal{O}_{m-1}, \\
\Omega_1 &= \mathcal{O}_0 \cap \mathcal{O}_1 \cap \ldots \cap \mathcal{O}_{m-2} \cap \mathcal{O}_{m-1}^c, \\
\vdots \\
\Omega_{2^m-1} &= \mathcal{O}_0^c \cap \mathcal{O}_1^c \cap \ldots \cap \mathcal{O}_{m-2}^c \cap \mathcal{O}_{m-1}^c.
\end{align*}
\]

(7.4)

Using $m$ Level Set functions $\phi_i$, associated to the domains $\mathcal{O}_i$, $i = 0, \ldots, m-1$, the domains $\Omega_j$, $j = 0, \ldots, 2^m-1$, are alternatively defined as:

\[
\begin{align*}
\Omega_0 &= \{x \in D, \ \phi_0(x) < 0, \ \phi_1(x) < 0, \ldots, \phi_{m-2}(x) < 0, \ \phi_{m-1}(x) < 0\}, \\
\Omega_1 &= \{x \in D, \ \phi_0(x) < 0, \ \phi_1(x) < 0, \ldots, \phi_{m-2}(x) < 0, \ \phi_{m-1}(x) > 0\}, \\
\vdots \\
\Omega_{2^m-1} &= \{x \in D, \ \phi_0(x) > 0, \ \phi_1(x) > 0, \ldots, \phi_{m-2}(x) > 0, \ \phi_{m-1}(x) > 0\}.
\end{align*}
\]
When it comes to describing the evolution of each phase $\Omega_j$, the velocity fields driving their motions unambiguously yield those $V_i$ driving the motions of the $\mathcal{O}_i$, $i = 0, \ldots, m - 1$. The Level Set Method can then be independently applied to each domain $\mathcal{O}_i$ along the lines of the previous sections.

**Remark 6.** Obviously, the number $M$ of described phases is not limited to a power of 2. In the general case, taking $m$ as the unique integer such that $2^{m-1} < M \leq 2^m$, and using $m$ domains $\mathcal{O}_i \subset D$, and $m$ associated level set functions $\phi_i$ in the way described above, one simply obtains the desired phases as reunions of the $\Omega_j$, $j = 0, \ldots, 2^m - 1$ defined in (7.4).

8. Other domain evolution methods

A wide literature exists around the numerical treatment of problems involving moving domains or interfaces. We now give a rough sketch of the main existing methods, which is deliberately clear-cut, and biased: in particular, this classification and the terminology used here are far from being uniform in the literature.

Throughout this section, $\Omega(t) \subset D$ denotes a moving domain in $\mathbb{R}^d$ with boundary $\Gamma(t)$, included in a larger, fixed computational domain $D$. Its motion is driven by the (vector) velocity field $V(t, x)$.

In the numerical setting, the time period $(0, T)$ of the evolution is, as usual, discretized into the sequence $t_n = n \Delta t$, $n = 0, \ldots, N := T/\Delta t$. Sometimes, for brevity, the explicit mention to time is dropped, and we shall simply refer to $\Omega$, $\Gamma$, $V$, etc.

8.1. Lagrangian methods.

The first great category is that of Lagrangian methods, sometimes also called moving mesh methods. They are possibly the most natural ones: at each step $t^n$ of the process, the domain $\Omega(t^n)$ (or, sometimes, its boundary $\Gamma(t^n)$ only) is equipped with a mesh $\mathcal{T}^n$. Hence, such methods are particularly relevant when the velocity field $V(t, x)$ driving the motion is dictated by physical properties, such as a fluid flow, and that its calculation demands accurate numerical simulations on $\Omega$ (see the bifluid flow example of Section 3).

From one step $t^n$ to the next one $t^{n+1}$, the mesh $\mathcal{T}^n$ is ‘moved’ into the new mesh $\mathcal{T}^{n+1}$, according to the velocity field $V^n := V(t^n, \cdot)$. For example, the most naive way to achieve this operation consists in transporting each node $x_i^n$ of $\mathcal{T}^n$ into a node $x_i^{n+1}$ of $\mathcal{T}^{n+1}$ as:

$$x_i^{n+1} = x_i^n + \Delta t V(t^n, x_i^n),$$
without altering the connectivity of the mesh. Unfortunately, doing so is likely to yield an invalid mesh, that is, a mesh in which some elements are overlapping; see Chapter ?? for precise definitions.

This operation of deforming a mesh according to a velocity field is complex in general (in particular, it is difficult to account for topological changes in the underlying domains in this way). Several heuristics can however help the process; see [27], Chap. 23, for more details around the issue of mesh deformation:

- An occasional remeshing of \( T^n \) - e.g. resampling its nodes, modifying its connectivities to eliminate very ill-shaped elements - may postpone the occurrence of difficulties in the mesh deformation process; on the contrary, excessive remeshing could cause a loss of computational efficiency and accuracy.
- The values of the velocity field \( V(t,x) \) are only relevant on the boundary of the moving domain \( \Omega(t) \); in particular, it is possible to modify \( V(t,x) \), retaining its values on \( \Gamma(t) \), so that it entails less distortion in the internal part of \( \Omega(t) \) (making the mesh deformation easier). The techniques used to achieve this have a lot in common with the velocity extension procedures described in Section 7.2.

Let us also mention that other Lagrangian techniques represent the evolving domain \( \Omega \) by means of a set of particles instead of a mesh; grossly speaking, \( \Omega \) is described by a set of points which are not connected to one another. The motion of \( \Omega \) is considerably simpler under this form. Of course, such techniques must come along with adequate numerical methods to perform operations on \( \Omega \), such as solving partial differential equations. See [37] for further details around those so-called meshfree methods.

The reader may consult [38, 61] for an overview and further discussions around Lagrangian methods.

8.2. Eulerian methods.

Unlike Lagrangian methods, Eulerian methods assume a fixed computational support, that is, a mesh \( G \) (often a Cartesian grid) of a computational domain \( D \). As the evolving domain \( \Omega \) is not explicitly discretized, some information about it has to be kept somehow. The nature of the retained information allows to distinguish two subclasses, namely front-tracking and interface-capturing methods.

8.2.1. Front-tracking methods.

Front-tracking methods rely on an explicit discretization \( T^n \) of the interface -or front- \( \Gamma(t^n) \) at every step \( t^n \) of the numerical process. This mesh may be obtained in several different ways:

- It may be maintained from the beginning, and transported from one step \( t^n \) to the next \( t^{n+1} \). This operation is achieved by relying on techniques very similar to those described in Section 8.1, in the case of Lagrangian methods. Like then, it may be accompanied with remeshing algorithms so that \( T^n \) stays well-shaped, and with some heuristics (which may use the computational grid \( G \)) to detect and resolve possible self-intersections.
- It may be reconstructed at each step \( t^n \), for instance from the datum of a set of markers (or particles) whose evolution is calculated.

Contrary to the case of Lagrangian methods, the velocity field \( V \) characterizing the motion is calculated by using the computational mesh \( G \) of \( D \), and not directly the mesh \( T^n \).

To set ideas, let us focus the discussion on the case where the calculation of \( V \) demands the resolution of a PDE posed on \( \Omega \) (we have in mind the example of Section 3). This PDE is solved on the Cartesian grid \( G \) at each step \( t^n \); hence, ways have to be found to account for the communication between this grid and the mesh \( T^n \) of the front. This can be done in several ways:

- Interpolation procedures may be devised from \( T^n \) into \( G \), to locate the position of \( \Omega(t^n) \) on \( G \), thus to allow for the calculation of related quantities (integrals on \( \Omega(t^n) \), curvature of \( \Gamma(t^n) \), etc...).
- A whole new mesh of \( \Omega(t^n) \) may be generated from \( G \) and \( T^n \). Of course, this procedure is very costly, and very dependent on the robustness of the used meshing algorithm.

The possibly best known implementations of front-tracking algorithms are those detailed in the articles [29, 62]. See also [33] for an interesting overview of complementary features.

Without getting into details, let us mention the Immersed Boundary method, introduced by Peskin in [47] (see also [41, 48] for overviews), which can be viewed as a front-tracking method in the particular context of
fluid-structure interactions. In a nutshell, the idea is to describe an elastic structure immersed in a fluid by using Lagrangian variables for the former part (a part which is numerically discretized and tracked thanks to Lagrangian techniques: moving mesh or particles), and Eulerian variables for the latter (a part which is numerically discretized on a fixed grid); the coupling between both parts is achieved owing to operations which are, in spirit, similar to those used in classical front-tracking methods.

8.2.2. Interface-capturing methods.

Interface-capturing methods account for the moving domain \( \Omega \) via auxiliary quantities, which are discretized on the fixed mesh \( \mathcal{G} \) of \( D \). The Level Set Method falls into this class, the evolution of \( \Omega \) being described by that of a Level Set function \( \phi \); at least two other techniques are worth mentioning.

8.2.2.1. Volume of Fluid Methods.

The Volume of Fluid Method (often abridged as VOF) was pioneered in the article [31], and has long been a reference method to account for the motion of a fluid domain, or several fluid phases.

The basic idea is to represent a fluid domain \( \Omega \) by a scalar function \( C \), which is constant by element \( T \in \mathcal{G} \) - \( T \) may be a e.g. a triangle, a square, depending on the nature of \( \mathcal{G} \). The quantity \( C_T \) stands for the ‘proportion of fluid’ present in the element \( T \), that is:

\[
C_T = \frac{1}{|T|} \int_{C_T} \chi_\Omega(x) \, dx,
\]

where \( \chi_\Omega \) is the characteristic function of \( \Omega \). In other words, \( C_T \) equals 1 if \( T \) is completely occupied by the fluid, 0 if it does not carry any fluid, and it takes an intermediate value if it is crossed by the interface \( \Gamma \).

A formal analysis reveals that this function \( C \) satisfies the evolution equation:

\[
\frac{\partial C}{\partial t} + V \cdot \nabla C = 0,
\]

which is to be understood in a very loose sense since \( C \) is constant par element! For this reason (and others), the numerical resolution of (8.1) is difficult and requires adequate numerical schemes. Most of those involve a reconstruction step, where the boundary of the fluid domain \( \Omega \) is inferred from the datum of \( C \).

As any interface-capturing method, the Volume of Fluid method naturally addresses the problem of topological changes. Furthermore, it has proved particularly successful in conserving the volume of \( \Omega \) in the case of an incompressible motion - a key feature in fluid simulations. However, the quality of the description of the boundary \( \Gamma \) is inherently less accurate than that featured by the Level Set method.

See [50, 51] for reviews of several aspects of the Volume of Fluid Method.

8.2.2.2. Phase-Field Methods.

The Phase Field Method was originally introduced in the context of two phases \( \Omega_0, \Omega_1 \), separated by an interface \( \Gamma \). The situation is described by a so-called order parameter, or Phase Field scalar function \( \phi \), which is discretized on the mesh of the computational domain \( D \) in the numerical setting. This function \( \phi \) assumes constant values in the bulk phases \( \Omega_0, \Omega_1 \) (typically \( -1 \) and \( 1 \)), and shows steep, yet smooth, variations in a thin tubular neighborhood of \( \Gamma \).

Instead of giving an abstruse presentation of the theoretical aspects of the method, let us concentrate the discussion on the particular example of the bifluid flow model of Section 3, whose notations are reused.

The various quantities characterizing the model (in our case, the density \( \rho \) and the viscosity \( \nu \) of the fluid) which are discontinuous at the interface \( \Gamma \), are approximated by quantities featuring a sharp, but smooth transition across this interface, for instance:

\[
\forall x \in D, \, \nu(x) \approx \nu(\phi(x)) := \frac{\nu_1 - \nu_0}{2} \phi(x) + \frac{\nu_1 + \nu_0}{2}.
\]

Thence, this phase field model is a smoothed approximations of the sharp-interface, exact model (3.1).
As far as it is concerned, the function $\phi$ is assumed to minimize a so-called *free energy* functional $J_\varepsilon(\phi)$, which is, for example, of the form:

\[
J_\varepsilon(\phi) = \int_{D} \left( \frac{1}{\varepsilon} W(\phi) + \frac{\varepsilon}{2} |\nabla \phi|^2 \right) \, dx.
\]

In this formula, $W$ is a so-called *double-well* potential, which assumes two minima at $\pm 1$. The meaning of this description is the following: $\phi$ is in a competition between minimizing the double-well $W(\phi)$ (which drives it towards a function taking only the values $\pm 1$, regardless of having sharp variations), and the gradient term, which constrains $\phi$ to retain some smoothness. The parameter $\varepsilon$ balancing both trends can be shown to be a measure of the thickness $\varepsilon$ of the smeared interface thus imposed by the method. Notice that, as $\varepsilon \to 0$, the constraint on the smoothness of the variations of $\phi$ is loosened in the expression (8.3).

The evolution of $\phi$ in time is driven by the minimization of the free energy $J_\varepsilon(\phi)$. Writing conservation laws to express this feature, in connection with the fact that $\phi$ is dragged by the fluid velocity $u(t, \cdot)$ of the phases leads to a diffusion equation for $\phi$, either of the *Allen-Cahn* or of the *Cahn-Illigard* type.

Depending on the point of view of the user, the Phase-Field Method can be seen as a theoretical and numerical artifice to mimic the sharp-interface formulation of the evolution problem at stake. It is then worth ensuring that the different choices (the free energy $J_\varepsilon(\phi)$, the evolution equation for $\phi$, the coupling between $\phi$ and $u$ expressed in equations such as (8.2), etc.) allow to retrieve the original model in some appropriate sense when the thickness parameter $\varepsilon$ goes to 0.

On the other hand, the Phase-Field Method can be thought of as a way to incorporate a complex, microstructural behaviors of the interface $\Gamma$ in a macroscopic model. In this case, the phase-field function $\phi$ is given a physical meaning, and the choices of the free energy and the diffusion equation for $\phi$ are based upon physical (notably thermodynamical) considerations.

From the numerical point of view, the Phase-Field Method has been successfully applied to several evolution problems such as, for instance, solidification [65] or multiphase flow [7]. It shares a lot of aspects with the Level Set Method: as a purely Eulerian method, it can handle large deformations (including topological changes) in a natural way; it may imply a stronger, physical coupling between the phase representation $\phi$ and the physical equations which lends itself to theoretical convergence analyses. However, the geometrical description of the interface $\Gamma$ is less accurate than in the context of the Level Set Method.

See [11, 13] for introductions to the Phase-Field Method.

### 8.3. Hybrid methods.

Numerous hybrid algorithms exist between Lagrangian and Eulerian methods, the perhaps most famous of which are the *Arbitrary Eulerian-Lagrangian* methods (ALE in short).

In Lagrangian methods, the computational mesh of the considered domain $\Omega(t)$ is transported in time according to the velocity field $V(t, x)$ of the motion. Doing so potentially offers maximum accuracy regarding the location of $\Omega(t)$, but this procedure may dramatically degrade the quality of the mesh, compromising the accuracy of computations. On the other hand, Eulerian methods use a fixed mesh of a computational domain $D$ at every step of the process.

As their name suggest, ALE methods go halfway between both points of view: the domain where calculations are carried out (and with it, the computational mesh) is moved in an arbitrary way, that is, independent from the exact position of $\Omega(t)$. The choice of a motion for this so-called *ALE domain* $\Omega_R(t)$, is dictated by a tradeoff between staying close from the deformed domain $\Omega(t)$, without jeopardizing with the quality of the attached mesh. The efficiency of an ALE method strongly depends on the devised strategy for the evolution of the ALE domain, and their description goes beyond the scope of this introduction.

To sum up, at a given time $t > 0$, three domains are to be considered:

- **The initial domain** $\Omega(0)$: in mechanical modeling, $\Omega(0)$ is referred to as the *material domain*, since it stands for the undeformed configuration, and contains the ‘particles’ $x$ of $\Omega$ at rest state.
- **The deformed domain** $\Omega(t)$, called *spatial domain* in mechanical modelling.

28
• The ALE domain $\Omega_R(t)$, or *reference domain*, which is the support of numerical computations.

The correspondance between the material and spatial domains $\Omega(0)$ and $\Omega(t)$ is described by the mapping:

$$\Omega(0) \ni x_0 \mapsto \varphi(t, x_0) = \chi(x_0, t, 0) \in \Omega(t)$$

where $t \mapsto \chi(x_0, t, 0)$ is the solution to (2.5) and represents the trajectory of a particle driven by the velocity $V(t, x)$, and lying at position $x_0$ at time $t = 0$ (see Section 2.2 and Chapter ??).

In turn, the motion of the ALE domain is described by the mappings

$$\Phi(t, \cdot) : \Omega_R(t) \to \Omega(0), \text{ and } \Psi(t, \cdot) : \Omega_R(t) \to \Omega(t),$$

which are explicitly given, once the moving strategy for the ALE domain has been set; see Figure 13.

![Figure 13. The three domains and meshes involved in an Arbitrary Lagrangian-Eulerian procedure, and the mappings for passing from one to another.](image-url)

In practice, all the theoretical and numerical calculations are performed on the ALE domain and mesh; hence, any quantity of interest attached to $\Omega(t)$ (normal vector, curvature, etc) must be transferred to $\Omega_R(t)$. Likewise, a partial differential equation posed on $\Omega(t)$ - for instance, in the model of Section 3, the bifluid Navier-Stokes equations - must be recast on the domain $\Omega_R(t)$: these are the so-called ALE formulations of these equations.

As far as ALE methods are concerned, we refer to the nice presentation [22] on this topic, and references therein for historical references and further discussions, or to the book [32].
9. Applications of the Level Set Method

Since its inception, the Level Set Method has become a reference method for capturing the motion of an evolving domain or interface, and it has found applications in very diverse fields. In this section, we mention some (actually very few) of these applications, with an obvious bias toward the knowledge of the authors.

9.1. Applications in Computational Fluid Dynamics.

The Level Set Method appeared in Computational Fluid Dynamics with the study of the motion of two compressible gases, separated by a sharp interface [42]. Soon after, it was used in [60] for describing the interface between two immiscible, incompressible fluids, driven by the Navier-Stokes bifluid equations. Since these seminal works, it has been a popular way for describing boundaries of domains filled with fluids, and many improvements and extensions of the original techniques have come out; see Chapter ??, or the review article [55] for more in-depth discussions.

9.2. Applications in shape optimization.

Shape optimization aims at finding the ‘best’ shape (or domain) $\Omega \subset \mathbb{R}^d$, i.e. that which minimizes a given functional $J(\Omega)$ depending on the domain itself. In the context that $\Omega$ stands for an elastic structure, one may imagine that $J(\Omega)$ is a measure of the stress developed in the structure, or (the opposite of) its fundamental frequency. If $\Omega$ is a pipe in which a fluid flows, $J(\Omega)$ may be for instance the dissipation of energy in the medium due to viscous effects.

Theoretical analyses make it possible to calculate explicitly a vector field $V_\Omega$ defined on the boundary $\Gamma$ of $\Omega$ - a so-called shape gradient -, such that, roughly speaking, moving $\Omega$ slightly in the direction $V_\Omega$ gives rise to a new shape with a smaller value of $J$.

Accordingly, a possible strategy for the minimization of $J(\Omega)$ consists in starting from an initial domain $\Omega(0)$, then tracking the evolution of the domain $\Omega(t)$, driven by this shape gradient $V(t,x) = V_\Omega(t)(x)$.

The Level Set method was introduced to represent such shape optimization problems in the articles [5, 56, 64], and we shall discuss this particular application in further details in Chapter ??.

9.3. Applications in image processing.

In the simplest mathematical models, a grey-scale image is represented by an intensity function $I : D \rightarrow \mathbb{R}$, where the working domain $D$ is the unit square $(0,1)^2$.

One key issue in image processing is that of segmentation: the image $I$ is generally composed of several areas with different contrasts, representing distinct objects. For instance, a medical image obtained by a process such as MRI identifies different tissues or organs by different intensities. Segmentation consists in capturing one of these particular regions, say $\Omega_T \subset D$, corresponding to a given intensity value $I_0$.

One method to achieve this was introduced in [12, 39], as a variant of the active contour model: a domain $\Omega(t)$ evolves in time (from an arbitrary initial position $\Omega(0)$) according to a velocity field $V(t,x)$ of the form:

$$V(t,x) = (f_0(x) + f_1(\kappa_t(x)))n_t(x),$$

where $f_0$ and $f_1$ are two scalar functions with the following meanings: $f_0$ ‘attracts’ the domain $\Omega(t)$ towards $\Omega_T$, while $f_1(\kappa_t(x))$ only involves the mean curvature $\kappa_t(x)$ of $\partial \Omega(t)$ and compels $\Omega(t)$ to stay ‘smooth enough’. In both aforementioned references, this evolution problem is numerically tackled with the Level Set Method.

Another major topic in image processing is denoising: in practice, the intensity function $I$ is often plagued with noise, and it is desirable to replace it with another one, say $\tilde{I}$, which is a smoothed version of $I$, but also retains the interfaces between the regions of different intensities. Several models have been proposed to give a mathematical flavour to this problem, among which the celebrated Mumford-Shah and Rudin-Osher-Fatemi (ROF) models. Both of them express the new image $\tilde{I}$ as the minimizer of an energy functional where the contours of the regions with different intensities appear explicitly. Again, the Level Set Method is a natural ingredient in a numerical method dedicated to such problems.
Let us eventually mention the work [26], which uses the level set method to tackle the stereo problem, that is the problem of reconstructing a three-dimensional scene, from the data of several two-dimensional views. More details around these issues, and many other examples of such problems in image processing may be found in the article [40] and the monograph [46].

References


